SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

November 2026, Vol. 69, Iss. 11, 210206:1–210206:16
https://doi.org/10.1007/s11432-026-5241-2

Special Topic: Intelligent Characterization and Modeling for Complex Systems

Discovering a compact meta-RL rule for critical node identification in complex networks

Zhenhua LIANG[1], Haotian WANG[1], Xueqiong LI[1*], Jun-Jie HUANG[1], Mingze QI[2], Hengzhu LIU[1] & Xinwang LIU[1]

[1]*College of Computer Science and Technology, National University of Defense Technology, Changsha 410073, China*
[2]*College of Science, National University of Defense Technology, Changsha 410073, China*

Received 15 June 2026/Revised 8 October 2026/Accepted 30 October 2026/Published online 16 November 2026

Abstract Identifying critical nodes whose removal fragments a complex network underpins applications across infrastructure resilience, epidemic blocking, and biological network analysis. Existing reinforcement learning (RL)-based graph learning methods train graph neural network (GNN) agents with hand-designed reinforcement-learning rules, so that every new task formulation, whether single-layer network dismantling, multilayer network dismantling, maximum independent set, minimum vertex cover, or maximum clique, requires re-engineering a reward or an architecture from scratch. We instead ask whether the RL update rule itself can be discovered once on the graph domain and re-used across this entire class of problems, by adapting a meta-RL paradigm in which a meta-network emits loss-shaping targets that train agents in place of a hand-designed RL loss. To our knowledge, this is the first systematic graph-domain instantiation of meta-learned RL update rules, realized through a graph-aligned meta-network. The proposed meta-network is compact and is meta-trained on five graph-combinatorial tasks on a single GPU in about 13 h, treating critical-node detection as a single underlying capability rather than as five disjoint tasks. GNN agents trained by this rule outperform all baselines on the established 53-network single-layer network dismantling benchmark; transfer zero-shot to 33 real-world multilayer networks, approaching the specialized state-of-the-art (SOTA) on interdependent networks and surpassing all baselines on multiplex networks; and approach the exact optimum on maximum independent set (MIS), minimum vertex cover (MVC), and maximum clique (MC) without per-task tuning. Because each task-specialized baseline leads only on its home protocol and degrades on the others, agents trained by our rule are the only ones that remain non-dominated across the single-layer, interdependent, and multiplex protocols simultaneously. Controlled comparisons under an identical training protocol attribute these gains to the discovered, graph-domain-specific update signal, rather than to model capacity or any classical RL update. For graph-combinatorial problems with short episodes and structure-driven action spaces, compact and domain-aligned meta-RL rules dominate large, domain-agnostic ones.

Keywords critical node identification, network dismantling, meta-reinforcement learning, graph neural networks, multilayer networks, complex networks

Citation Liang Z H, Wang H T, Li X Q, et al. Discovering a compact meta-RL rule for critical node identification in complex networks. Sci China Inf Sci, 2026, 69(11): 210206, https://doi.org/10.1007/s11432-026-5241-2

1 Introduction

The function of many complex systems depends on a small set of *critical nodes*. Their targeted removal disproportionately influences the system's collective behavior. A canonical example is the spread of an epidemic over a contact network. A small fraction of high-degree super-spreaders is responsible for the majority of secondary infections. Identifying them ahead of time for vaccination or quarantine is the most cost-effective non-pharmaceutical intervention known [1].

The most basic abstraction of this phenomenon is single-layer *network dismantling*: in what sequence should nodes be removed to fragment the network most rapidly? A single layer, however, cannot capture two relations that pervade real-world systems. The first is *interdependence*: nodes in one layer rely on nodes in another, so a local failure can cascade across coupled layers. The blackout of September 2003 in Italy is a canonical instance, where a handful of failed power stations disabled the communication layer that served them, which in turn tripped further stations and left tens of millions without electricity [2]. The second is *multiplexity*: the same nodes interact through several kinds of relations at once, and one relation can substitute for another when it is lost. Dismantling therefore has two multilayer forms—one that must anticipate cascades, and one that must break connectivity that any layer alone can sustain.

* Corresponding author (email: lixueqiong13@nudt.edu.cn)

© Science China Press 2026

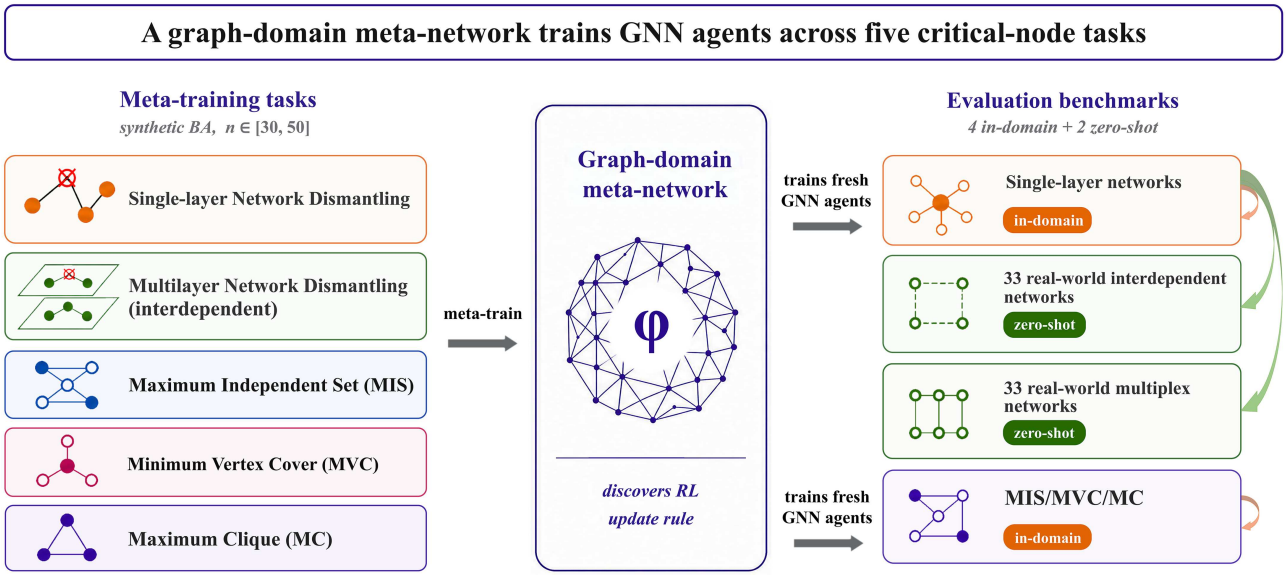

Figure 1 (Color online) Overview of the proposed discovered reinforcement-learning update rule for the graph domain. A single meta-network, meta-trained on five critical-node tasks, thereafter trains GNN agents on each task in place of a hand-designed RL loss. The single-layer network dismantling agent is evaluated in-domain on 53 real-world networks and transfers zero-shot across protocols to interdependent and multiplex networks, while agents for MIS, MVC, and MC reach near-optimal solutions without per-task tuning. The orange and green arrows indicate in-domain evaluation and zero-shot cross-protocol transfer, respectively. Multiplex networks present the strictest transfer test: they appear neither in agent training nor in meta-training.

The same search for critical nodes underlies three classical graph problems. Maximum independent set (MIS) asks for the largest set of nodes with no edge between them; its complement, minimum vertex cover (MVC), asks for the smallest set of nodes that touches every edge; and maximum clique (MC) asks for the largest set of nodes that are all mutually adjacent. Each is, at heart, a question about which nodes are structurally decisive. These five problems look different but share one core question—where are the critical nodes—and all are nondeterministic polynomial-time (NP)-hard on general graphs [3]. Exact solutions are therefore out of reach at scale, which motivates learning-based methods.

The dominant learning-based approach, introduced by FINDER [4], treats each of these problems as a separate Markov decision process. It trains a graph neural network (GNN) with a hand-designed reinforcement learning (RL) update rule: n-step Q-learning, REINFORCE [5], or an evolutionary-RL hybrid. Every move to a new task formulation forces a re-engineered reward shape or a new network architecture from scratch. From single-layer to multilayer, from giant connected component (GCC) to pairwise connectivity, or from dismantling to MIS/MVC/MC, each requires a fresh design. The critical-node detection ability that underlies all of these tasks is therefore fragmented across method-specific implementations. It is not expressed as a single shared capability. We instead take the position that the RL update rule itself should be discovered once on the graph domain and re-used across all five tasks.

Our starting point is the observation that, although every method above hand-designs its update rule, the recently proposed paradigm of meta-learning the update rule itself [6–8] makes this commitment optional. A meta-network is trained once to emit loss-shaping targets that train agents. It takes the place of a hand-designed RL loss. To date this paradigm has been instantiated and validated only in the video-game domain. We adopt it for the graph domain because it offers exactly what the fragmentation problem above lacks: a single shared object, the update rule, that all five critical-node tasks can reuse.

In this paper we introduce the first discovered reinforcement-learning update rule for the graph domain, to our knowledge (Figure 1). A single meta-network is meta-trained once on five critical-node tasks: single-layer network dismantling, multilayer network dismantling, MIS, MVC, and MC. It thereafter trains GNN agents on each of them in place of any hand-designed RL loss. The premise behind treating all five problems as a single mixture is that critical-node detection is a single underlying ability. Rather than re-engineering a reward, an architecture, or a heuristic per task, we let one shared update rule express this ability across all five tasks.

Importantly, our method is not transfer learning at the agent level. Each agent is still trained from scratch on its target task. What is shared across tasks is the discovered RL update rule that trains that agent. This separation makes the source of any performance gain isolable. Update rules (REINFORCE, PPO, a general-purpose meta-RL

rule trained on video games, and the proposed method) can be compared under an otherwise identical protocol: the same agent backbone, training distribution, seeds, and loss-coefficient defaults. Remaining differences are then attributable to the update rule alone.

Under the fair protocol described above, this paper makes the following contributions.

• (C1) A discovered RL update rule for the graph domain. We meta-train, to our knowledge for the first time, a graph-domain meta-network that emits the RL loss-shaping targets used to train GNN agents on critical-node tasks. A single meta-training run on five problems—single-layer network dismantling, multilayer network dismantling, MIS, MVC, and MC—yields one shared update rule that subsequently trains an agent for any of them.

• (C2) Empirical gains across single-layer, multilayer, and cross-task benchmarks. Agents trained with our rule (1) outperform all baselines on the established single-layer benchmark comprising 53 real-world networks in [9]; (2) transfer zero-shot to 33 real-world multilayer networks of two types, approaching the specialized state-of-the-art (SOTA) on interdependent networks and surpassing all baselines on multiplex networks, and remaining close to the specialized SOTA when the interdependent networks are scaled up by an order of magnitude; and (3) come within $\approx$1% of the exact optimum on MIS, MVC, and MC without per-task tuning. A strict fair-protocol ablation attributes this advantage to the discovered graph-domain RL signal itself, not to capacity or to any classical RL update.

• (C3) An interpretation of what the discovered rule has learned. A prediction-semantics analysis shows that the meta-network teaches the agent to encode critical-node detection directly into its policy distribution: the agent's policy is sharply peaked when it is about to remove a structurally critical node, and quasi-uniform on routine steps. A control comparison confirms this detection signal is specific to the discovered graph-domain rule rather than a generic by-product of RL training, since neither a general-purpose meta-RL rule trained on video games nor the hand-designed rules reproduce it. This is the graph-domain analog of the prediction-semantics finding reported for video-game environments and provides a concrete reading of what a graph-domain meta-RL update rule has discovered to teach.

Section 2 reviews related work. Section 3 formalizes the five tasks. Section 4 describes our meta-network and agent training. Section 5 states the fair-protocol setup. Section 6 reports the three contributions. Sections 7–8 discuss limitations and conclude.

2 Related work

2.1 Single-layer network dismantling

Network dismantling (ND) seeks an ordered sequence of nodes whose iterative removal minimizes a connectivity measure of the residual graph. The minimum-set formulation is NP-hard on general graphs. Classical methods include belief-propagation solvers, the closed-form CoreHD rule [10], and explosive percolation-based decycling [11]. Centrality-based attacks—high-degree adaptive (HDA) [12–14], high-betweenness adaptive (HBA), HPRA, collective influence (CI) [1], and k-shell decomposition [15,16]—compute a node score on the current graph and iteratively remove its top-ranked node; they are fast and often competitive on real-world networks. Recent work extends to spatial attributes [17] and directed systems [18].

FINDER introduced a deep reinforcement learning formulation of ND that has become the standard learning baseline. Each removal is an action in a Markov decision process, the state is the current residual graph, and the reward is the decrease of the dismantling objective. The agent jointly learns a node embedding via GraphSAGE-style message-passing and a deep Q-network that scores candidate removals. FINDER is trained on small synthetic networks and deployed online to large real-world networks via a 1%-batch selection strategy. It significantly outperformed classical heuristics and demonstrated that an RL agent trained only on synthetic graphs could generalize to large real-world networks.

Subsequent work extends this paradigm: replacing handcrafted features with attention profiles to scale to millions of nodes [19], continuous relaxation for efficiency [20], graph contrastive representations [21], and neural-algorithmic strategies [22]. All such methods share two limitations: (1) they train task-specific policies from scratch, and (2) the underlying RL update rule is hand-designed. The third-party benchmark in [9] provides a uniform 10%-GCC-threshold protocol on real-world networks.

2.2 Multilayer network dismantling

Coupled and multilayer networks have become a central object of study in complex-network research [23, 24]. Existing multilayer methods fall into three categories. Centrality-based: HDA selects nodes by maximum degree

across layers [12, 14]; effective multiplex degree (EMD) [25] is an iterative weight-propagation rule, among the strongest non-learned baselines. These are fast but cannot react to evolving topology. Heuristic and optimization: combinatorial search or evolutionary approaches perform well on small-to-medium networks but scale poorly. Learning-based: early work on multiplex representation learning (MANE, MGCN, mGNN, Multi-GCN) targeted node classification or link prediction rather than dismantling, building on the broader graph-neural-network toolkit of inductive embeddings (e.g., GraphSAGE [26], graph attention [27], and temporal GNN forecasting [28]) and on the line of work applying deep reinforcement learning (DRL) to combinatorial optimization over graphs (e.g., S2V-DQN [29]). More recent papers bring this machinery to the dismantling task: early FINDER-style multiplex extensions aggregate all layers into a single supra-graph and generally lose explicit intra- versus inter-layer signals, while semi-supervised GNN-DRL formulations targeting interdependent networks tend to operate transductively and do not transfer zero-shot to unseen MLNs. The current SOTA on interdependent networks is MultiDismantler (MD) [30]: a DRL method trained directly on synthetic multilayer Barabási-Albert (BA) interdependent graphs ($n_{\text{per}} = 30$–50 per-layer nodes) with a domain-specific value function and reward, which currently achieves the lowest published mean accumulated normalized connectivity on real-world interdependent networks under both unit-cost and node-degree-weighted removal regimes. All learning-based MLN dismantling methods to date are trained directly on multilayer environments and on one specific problem type (interdependent or multiplex), leaving zero-shot cross-type transfer and the resulting gap to specialized SOTA unexplored.

2.3 Reinforcement learning: from hand-designed to discovered update rules

The way an RL agent updates its parameters has historically been hand-designed by humans [6, 31]. Successive landmarks have expanded what an agent predicts and how the loss is shaped: temporal-difference (TD) learning targets bootstrapped scalar returns; Q-learning targets action values, exemplified by DQN reaching human-level Atari play [32]; distributional and categorical losses (C51, QR-DQN) target return distributions rather than expectations; PPO [33] adds a clipped policy ratio to stabilize on-policy updates; auxiliary tasks (UNREAL), successor features, and auxiliary-control objectives provide additional representation-shaping losses; and Retrace, V-trace, and IMPALA correct for off-policy distribution shift. Each advance broadened the space of prediction targets an agent uses to update, but in every case the prediction semantics, whether value, return distribution, or advantage, were chosen by the designer, and only the hyper-parameters (learning rate, target update period, loss-coefficient ratios) were tuned automatically.

DiscoRL [6] takes the final step in this progression: it replaces the hand-designed target itself with a meta-learned target. A meta-network emits target distributions $(\hat{\pi}, \hat{y}, \hat{z})$ that the agent minimizes a KL divergence against, where the semantics of the prediction vectors y and z are not pre-specified but discovered during meta-training. The meta-network is trained by meta-gradient over a population of agents interacting with diverse environments. The largest released variant, Disco103, contains 754k parameters meta-trained on 103 Atari, ProcGen, and DMLab-30 environments—all video games—and transfers to several non-Atari benchmarks including network dismantling, outperforming manually-designed RL rules.

No discovered update rule yet exists for the graph domain. Every RL-based graph method above hand-designs its update rule, and DiscoRL—the only discovered alternative—was neither designed nor meta-trained for graphs: its 103 video-game environments share no obvious inductive bias with combinatorial optimization on discrete structures, and its 600-dimensional prediction targets are sized for Atari-scale value functions rather than the short, structure-driven episodes of graph tasks. Reproducing its discovery process, moreover, costs hundreds of TPU-days, beyond the reach of a typical academic GPU budget. And its analysis of what a discovered rule learns to predict was carried out only for video-game agents, leaving open whether a graph-domain rule would acquire an interpretable, task-aligned prediction structure.

3 Preliminaries

We consider five graph-optimization tasks: single-layer network dismantling, multilayer network dismantling, maximum independent set (MIS), minimum vertex cover (MVC), and maximum clique (MC). Each task is formulated as sequential node selection, but the meaning of a selection and the terminal objective depend on the task. Dismantling seeks to reduce network connectivity with as few removals as possible, whereas MIS, MVC, and MC construct feasible node subsets with optimal cardinality.

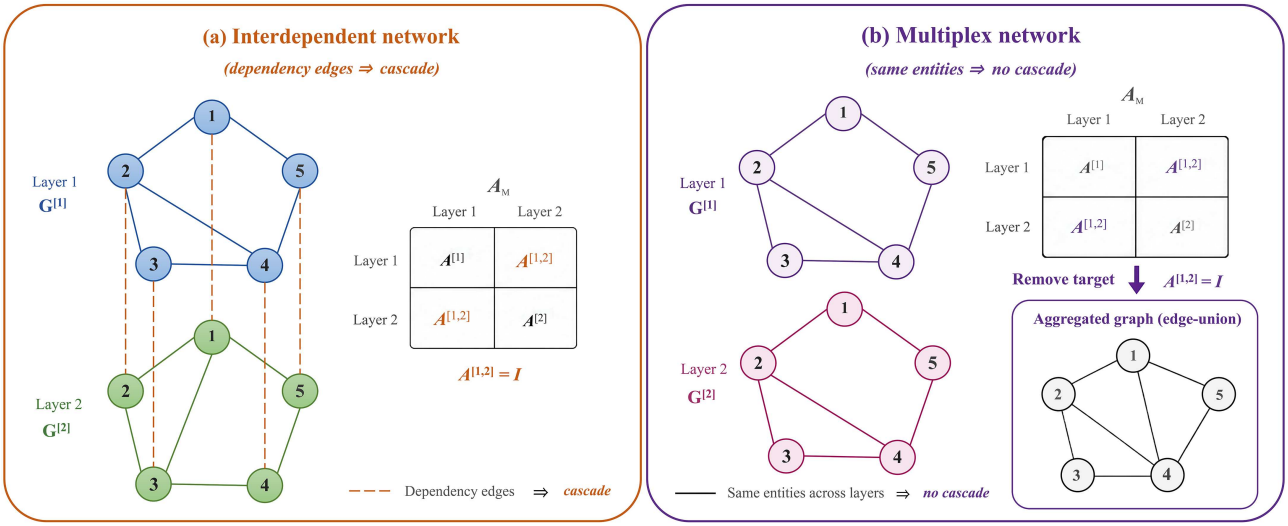

Figure 2 (Color online) Two multilayer network dismantling formulations. (a) In an interdependent network, one-to-one coupling encodes dependency and permits cascading failures. (b) In a multiplex network, coupling only aligns the same entity across layers; there are no dependency-induced cascades, and dismantling is evaluated on the edge-union graph. Although both coupling blocks can be written as the identity after relabeling, their operational meanings differ.

3.1 Single-layer network dismantling

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph with $N = |\mathcal{V}|$ nodes, and let $\sigma(\mathcal{G})$ denote the size of its largest connected component, referred to as the GCC. For a removal order $(v_1, \ldots, v_N)$, define $V_{1:k} = \{v_1, \ldots, v_k\}$. The accumulated normalized connectivity (ANC) over a complete removal sequence is

$$R(v_1, \ldots, v_N) = \frac{1}{N} \sum_{k=1}^{N} \frac{\sigma(\mathcal{G} \setminus V_{1:k})}{\sigma(\mathcal{G})}. \tag{1}$$

A lower ANC means that the removal sequence fragments the network earlier.

Our experiments use a thresholded numerical version of this area-under-the-connectivity-curve metric. For a connectivity trajectory $c_0, c_1, \ldots$, let

$$K_\tau = \min\{k : c_k \leqslant \tau c_0\}, \qquad R_\tau^{\text{eval}} = \frac{1}{N^2} \operatorname{Simpson}\left(\{(k, c_k)\}_{k=0}^{K_\tau}\right). \tag{2}$$

Here the horizontal axis counts deliberate node removals, and Simpson denotes numerical integration of the observed connectivity trajectory. We use $\tau = 0.1$ throughout unless stated otherwise. Eq. (2) is the reported experimental ANC; it differs from (1) through early stopping, numerical integration, and, when $c_0 \neq N$, normalization. For initially connected graphs, $c_0 = N$ and the normalizations coincide.

3.2 Multilayer network dismantling

We study two-layer networks with $N \equiv n_{\text{per}}$ aligned entities per layer, where n_{per} denotes the per-layer node count. Let $\mathcal{M} = \{\mathcal{G}, \mathcal{C}\}$, where $\mathcal{G} = \{G^{[1]}, G^{[2]}\}$ contains the intra-layer graphs and $\mathcal{C}$ specifies inter-layer coupling. The block adjacency representation is

$$A_{\mathcal{M}} = \begin{pmatrix} A^{[1]} & A^{[1,2]} \\ A^{[2,1]} & A^{[2]} \end{pmatrix}, \qquad A^{[2,1]} = (A^{[1,2]})^\top. \tag{3}$$

The same identity coupling can represent either dependency between distinct layer-specific nodes or alignment of the same entity across layers. These interpretations induce different dismantling objectives (Figure 2).

Interdependent networks. Removing a node disables its dependent partner in the other layer. Nodes disconnected from their layer's GCC also fail, and dependency and connectivity failures are iterated until convergence. Connectivity is measured by the giant mutually connected component (GMCC),

$$\Gamma_{\text{GMCC}}(\mathcal{M}) = \max_i |C_i^{\text{mut}}|, \tag{4}$$

where C_i^{mut} is a maximal set of aligned entities that is connected within each layer and satisfies the dependency constraints. Component sizes count aligned entities rather than summing their copies across layers.

The full-sequence objective is

$$R_{\mathrm{GMCC}}(V_{1:N}) = \frac{1}{N}\sum_{k=1}^{N}\frac{\Gamma_{\mathrm{GMCC}}(\mathcal{M}\setminus V_{1:k})}{\Gamma_{\mathrm{GMCC}}(\mathcal{M})}, \tag{5}$$

where the residual network is evaluated after cascade convergence. Experimentally, we use (2) with $c_k = \Gamma_{\mathrm{GMCC}}(\mathcal{M}\setminus V_{1:k})$ and normalization n_{per}^2. Only deliberate removals advance the horizontal axis; failures induced by a cascade contribute to the resulting connectivity drop.

Multiplex networks. The layers share entities but have no inter-layer dependency failures. The dismantling target is the aggregated edge-union graph $\mathcal{A} = (\mathcal{V}, \mathcal{E}_{\mathrm{agg}})$, whose adjacency is

$$a_{ij}^{[\mathrm{agg}]} = \begin{cases} 1, & \exists\, \ell \in \{1,2\} : a_{ij}^{[\ell]} = 1, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

Its giant aggregated connected component (GACC) is

$$\Gamma_{\mathrm{GACC}}(\mathcal{A}) = \max_{C_i \in \mathrm{CC}(\mathcal{A})} |C_i|. \tag{7}$$

The reported ANC follows (2) with $c_k = \Gamma_{\mathrm{GACC}}(\mathcal{A}\setminus V_{1:k})$. Thus, interdependent and multiplex network dismantling differ in the connectivity dynamics being optimized, not merely in their adjacency representation.

3.3 MIS, MVC, and MC

The remaining tasks construct a subset $\mathcal{S} \subseteq \mathcal{V}$:

- **MIS** maximizes $|\mathcal{S}|$ subject to no two selected nodes being adjacent;
- **MVC** minimizes $|\mathcal{S}|$ subject to every edge having at least one endpoint in $\mathcal{S}$;
- **MC** maximizes $|\mathcal{S}|$ subject to every pair of selected nodes being adjacent.

All three problems are NP-hard. To make the direction of comparison consistent, we report

$$f_k(\mathcal{S}) = \begin{cases} 1 - |\mathcal{S}|/N, & k \in \{\mathrm{MIS}, \mathrm{MC}\}, \\ |\mathcal{S}|/N, & k = \mathrm{MVC}. \end{cases} \tag{8}$$

Lower values are better. Where an exact solution is available, we additionally evaluate the normalized optimality gap $f_k(\mathcal{S}) - f_k^\star$.

Maximum clique enumeration provides the exact MC optimum. MIS is obtained by solving MC on the complement graph, and MVC follows from the complement of a maximum independent set. These reductions allow exact references on the small graphs used in our cross-task evaluation.

4 Method

Our method learns an update rule for training graph agents, rather than a single policy shared by all tasks (Figure 3). A graph-domain meta-network ϕ produces loss-shaping targets for a task agent π_θ. During meta-training, ϕ is optimized through the learning trajectories of freshly initialized agents. After meta-training, ϕ is frozen and used to train new task-specific agents.

This separation serves two purposes. First, it allows a common learning rule to operate across tasks with different objectives and reward structures. Second, it separates transfer of the update rule to fresh agents from transfer of a trained policy to unseen graphs or dismantling protocols.

4.1 Graph-agent interface

The agent operates on the residual graph. Node features include an alive/removed indicator, current degree, clustering coefficient, k-shell index, and normalized eigenvector centrality. A message-passing GNN produces node embeddings from which the agent computes a policy over currently admissible actions. For dismantling, an action

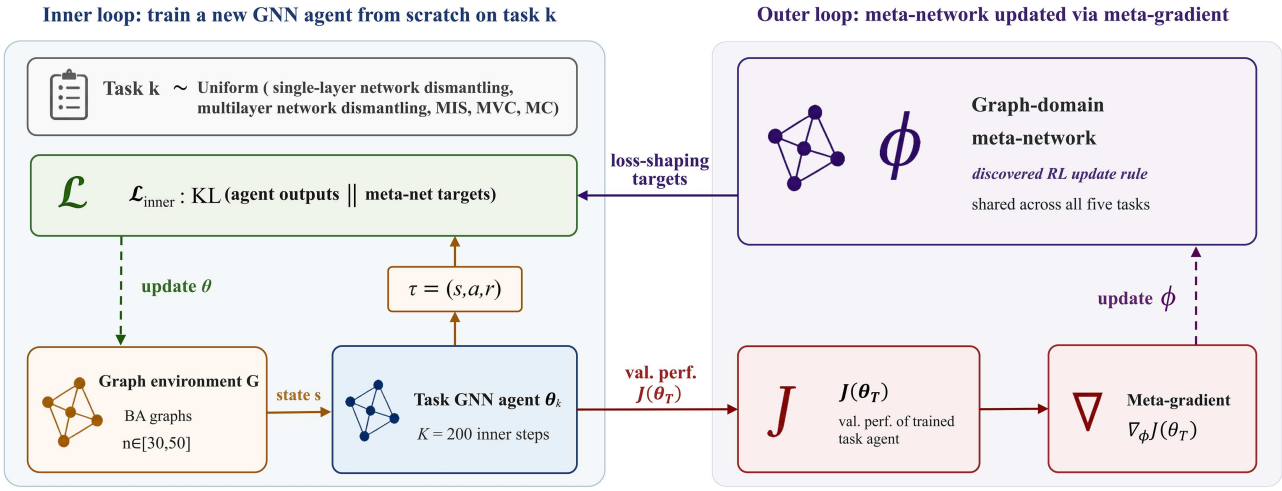

Figure 3 (Color online) Two-loop training framework. The inner loop trains a freshly initialized GNN agent using distributional targets supplied by the meta-network. The outer loop updates the meta-network through the agent's validation return using truncated meta-gradients. Once meta-training is complete, the meta-network is frozen and reused to train task-specific agents. Real-world test graphs are excluded from training.

removes an alive node. For MIS, MVC, and MC, the task environment interprets node selections according to the corresponding feasibility constraints.

Alongside the action policy π, the agent produces two learned prediction distributions $y, z \in \mathbb{R}^p$, a categorical action-value distribution q, and an auxiliary policy p_{aux}. The semantics of y and z are not prescribed: the meta-network learns how to use them to shape subsequent agent updates. The categorical value representation supplies a distributional training signal rather than a scalar regression target.

The graph encoder uses permutation-equivariant message passing, and action masking accommodates a changing number of available nodes. This interface lets the same meta-network process learning trajectories from different graph tasks without requiring fixed-size graph observations.

4.2 Graph-domain meta-network

We use SimplifiedMetaNet, a compact recurrent meta-network with approximately 155k parameters. It receives trajectories of agent outputs, rewards, and termination indicators, but no explicit task label. Three design choices adapt the architecture to short, structure-driven graph episodes.

Compact prediction space. The prediction dimension is $p = 128$. A compact output space limits the cost of generating and matching distributional targets during repeated inner-loop updates.

Trajectory-level recurrence. A trajectory LSTM with hidden dimension 128 summarizes within-episode feedback. This allows the targets to depend on the sequence of decisions and rewards rather than only on the current prediction.

Learning-lifetime recurrence. A meta-RNN cell with hidden dimension 64 processes batch-time-averaged inputs and multiplicatively gates the trajectory representation. Its state persists across inner updates, allowing the loss-shaping rule to respond to the agent's learning history. This distinguishes the proposed adaptive target generator from a static auxiliary loss. Component-level parameter counts and recurrent-state ablations are provided in Appendix M.

4.3 Distributional inner-loop training

For a rollout indexed by t, the meta-network supplies loss-shaping targets for the policy and prediction heads. The agent minimizes

$$\begin{aligned}\mathcal{L}_\theta = {} & \alpha_\pi D_{\text{KL}}(\pi \| \hat{\pi}) + \alpha_y D_{\text{KL}}(y \| \hat{y}) + \alpha_z D_{\text{KL}}(z \| \hat{z}) \\ & + \alpha_{\text{aux}} D_{\text{KL}}(p_{\text{aux}} \| \pi^{s'}) + \alpha_v H(\hat{q}, q).\end{aligned} \tag{9}$$

The loss is averaged over the sampled rollout. Here $\pi^{s'}$ is the next-state policy used for auxiliary self-distillation, and $\hat{q}$ is the categorical, two-hot-encoded TD target used by the value-learning component. The term $H(\hat{q}, q)$ is categorical cross-entropy, equivalently $D_{\text{KL}}(\hat{q} \| q)$ up to target entropy; it is not a scalar mean-squared error.

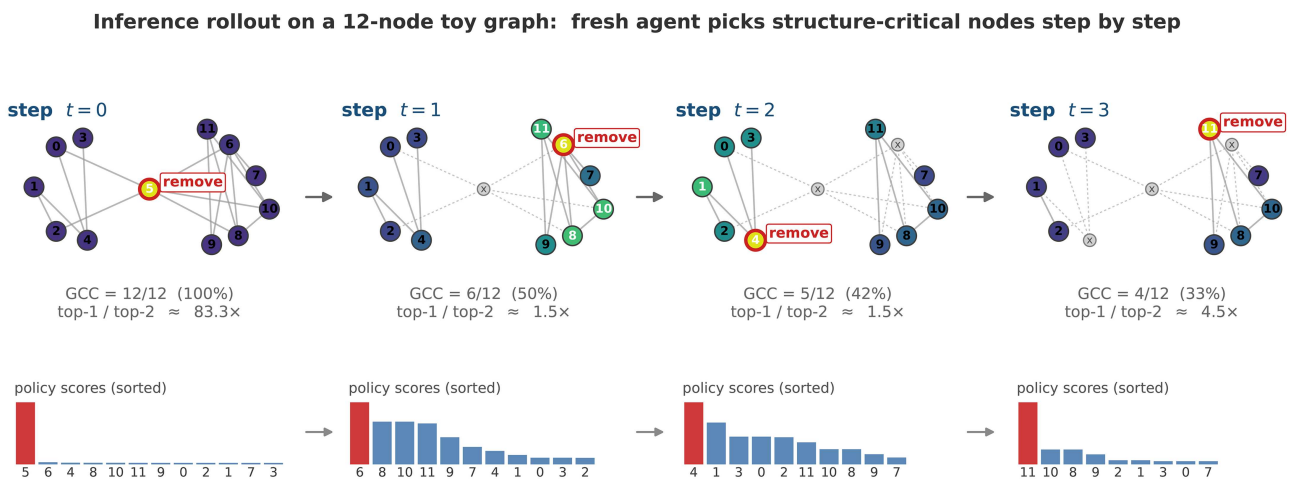

Figure 4 (Color online) Inference-time rollout on a 12-node toy graph that illustrates the qualitative behavior of the trained rule. At each step t the color of every alive node encodes the task agent's softmax policy score (viridis, brighter = higher); the red-outlined node is the top-1 choice that the rule is about to remove; faded markers indicate previously removed nodes (retained in the layout to make structural change easy to follow). Bottom-row bars show the sorted policy scores. Step $t = 0$ identifies a structural cut (node 5, the unique bridge between the two communities), collapsing the giant component from 12 to 6 in a single removal; subsequent steps target the remaining sub-hubs.

Rewards express task progress through a shared trajectory interface. Dismantling uses normalized connectivity reductions, while the combinatorial tasks use terminal objective feedback with signs chosen consistently with maximization of return. A fixed per-task baseline is used as a control variate in the return-based gradient estimates.

4.4 Meta-training and deployment

Let U_ϕ denote an agent update obtained from (9). Starting from $\theta_0 \sim \rho$, the inner loop performs

$$\theta_{j+1} = U_\phi(\theta_j, \xi_j), \qquad j = 0, \ldots, K-1, \tag{10}$$

where ξ_j is a rollout batch from task k. The outer objective maximizes the trained agent's expected validation return

$$\phi^\star = \arg\max_\phi \mathbb{E}_{k\sim\mathcal{T}} \mathbb{E}_{\theta_0\sim\rho} \left[J_k \left(U_\phi^K(\theta_0) \right) \right], \tag{11}$$

where $\mathcal{T}$ gives equal weight to the five tasks.

Fresh agent initialization prevents the outer loop from optimizing only the continuation of a particular policy. Multi-task training instead encourages targets that support learning under different connectivity and cardinality objectives.

We optimize (11) using truncated meta-gradients. Only the final M inner updates are differentiated through; earlier agent states are detached. This reduces the memory required for the differentiable unroll while retaining dependence on recent learning dynamics. The recurrent meta-state summarizes the preceding learning history, although truncation does not recover gradients through the detached updates.

After meta-training, ϕ is frozen. A new task agent is trained from scratch for each task and seed using the same target-generation interface. At test time, the trained agent selects the highest-probability admissible node and recomputes graph embeddings after each state transition. There is no look-ahead search, test-instance fine-tuning, or further update to ϕ. Figure 4 illustrates a 4-step rollout on a 12-node toy graph.

5 Experimental setup

5.1 Training configuration and comparison protocol

Training data and optimization. Meta-training uses synthetic BA graphs. Single-layer dismantling, MIS, MVC, and MC use small graphs; interdependent network dismantling uses two-layer BA networks to limit cascade simulation cost. Multiplex aggregation is not a meta-training task. No real-world test graph is used for training or model selection.

Each outer step batches 32 task-agent pairs, with $K = 200$ inner updates per agent. The outer optimizer is Adam with learning rate 3×10^{-4} for 200 steps. Per-task control-variate baselines are estimated once from 128 synthetic

validation graphs and held fixed. The task agent uses three message-passing layers with embedding dimension 64 and hidden dimension 128. The loss coefficients $(\alpha_\pi, \alpha_y, \alpha_z, \alpha_{\text{aux}}, \alpha_v) = (1, 1, 1, 1, 0.2)$. Experiments use one A800 GPU and PyTorch 2.4.

Comparison with Disco103. Disco103 uses the authors' released meta-network weights. Both frozen meta-networks train fresh task agents using the same GNN encoder, graph distributions, loss coefficients, seed protocol, and validation-selection procedure. Prediction heads retain each teacher's native width: $p = 128$ for Ours and $p = 600$ for Disco103, without truncation or projection. Comparisons should be interpreted as comparisons of the resulting update-rule systems, not as isolating one architectural factor.

5.2 Evaluation data and baselines

All dismantling methods are scored using the same thresholded ANC definition in (2) under the relevant connectivity dynamics. Heuristic configurations are detailed in Appendix I.

Single-layer dismantling. We evaluate 53 real-world networks spanning BIO, INFO, SOCIAL, and TECH. Of these, 49 come from the curated benchmark in [9]; Crime, HI-II-14, Enron, and Facebook are added from the FINDER benchmarks. Baselines are HDA, CI with $\ell = 2$, CoreHD, GND, FINDER, and Disco103. To test node-scale generalization, we additionally evaluate trained agents on BA ($m = 4$) graphs at three scales.

Interdependent and multiplex network dismantling. We use 33 real-world multilayer datasets from SOCIAL, TECH, BIO, and ECONOMIC domains, with $n_{\text{per}} \in [13, 1043]$. Each dataset is evaluated under both the GMCC and GACC protocols.

Ours and Disco103 use single-layer-trained task agents without additional multilayer training; this is agent-level zero-shot transfer. GMCC baselines are HDA, EMD, MD, and Disco103. GACC evaluation additionally includes FINDER; HDA and FINDER operate on the aggregated graph. Three additional genetic multilayer networks with $n_{\text{per}} \in \{668, 1358, 2324\}$ are evaluated in Appendix O.

MIS, MVC, and MC. Fresh agents are trained with the frozen meta-network and selected by the best-of-five-seed validation protocol. Evaluation uses 20 BA graphs at the training scale and 20 graphs at each of $n = 100$ and 200, without retraining. Comparisons include per-task greedy heuristics, a random agent, Disco103, and exact optima computed via the reductions in Subsection 3.3.

Ablations. We compare the full method with single-task graph meta-training (single-task), multi-task REINFORCE (MTR), single-task REINFORCE (STR), and Disco103, using the 53-network single-layer and 33-network GMCC benchmarks.

For policy diagnostics, we analyze 10 BA test graphs with $n \in \{30, 100, 500\}$, comprising 511 steps. Removals in the top 10% of GCC drops are *salient*; those in the bottom 50% are *routine*. We define the dispersion-based detection statistic

$$D = \Pr(d_{\text{salient}} < d_{\text{routine}}) + \tfrac{1}{2}\Pr(d_{\text{salient}} = d_{\text{routine}}), \tag{12}$$

where d_t is the standard deviation of node scores over alive nodes at step t. This measures separation of score dispersion between the two step classes; it is not a calibrated measure of policy confidence. The same analysis is applied to REINFORCE, PPO, and Disco103 controls.

6 Results

6.1 Single-layer dismantling on real-world networks

Ours achieves the lowest mean ANC on the 53-network benchmark, with an overall score of 0.0878 (Table 1). It leads INFO, SOCIAL, and TECH and ranks second on BIO, where Disco103 is better by 0.0024. Ours also wins on a majority of networks against every baseline, indicating that the aggregate advantage is not limited to a small number of favorable instances.

The reported Wilcoxon tests give $p = 3.7 \times 10^{-4}$ against Disco103 and $p = 4.4 \times 10^{-3}$ against HDA. The corresponding bootstrap 95% confidence intervals for the mean improvement are $[0.00067, 0.0079]$ and $[0.0010, 0.0056]$, both excluding zero. Figure 5 shows the network-level comparison with Disco103.

The advantage is not universal: Ours trails HDA by at least 0.01 on cfinder-google and power-eris1176, both of which diverge substantially from the BA training distribution. Thus, the results support broad empirical generalization from small synthetic training graphs, rather than dominance on every topology. Appendix J reports that the overall ranking remains favorable across nine combinations of integration rules and stopping thresholds.

Table 1 Mean ANC on 53 single-layer networks. Lower is better; best values are bold and second-best values underlined. The last column reports Ours' wins/losses against each baseline.

Method	BIO_{10}	INFO_{12}	SOCIAL_{16}	TECH_{15}	Overall_{53}	Ours' W/L
Ours	0.1499	**0.0892**	**0.0811**	**0.0526**	**0.0878**	–
Disco103	**0.1475**	0.0989	0.0835	0.0583	0.0919	42/11
HDA	0.1539	0.0919	0.0851	0.0551	0.0911	35/18
FINDER	0.1539	0.0916	0.0851	0.0583	0.0919	36/17
CI	0.1533	0.1046	0.0841	0.0551	0.0936	36/17
CoreHD	0.1601	0.1111	0.0911	0.0594	0.0997	49/4
GND	0.1777	0.1246	0.1098	0.0782	0.1170	51/2

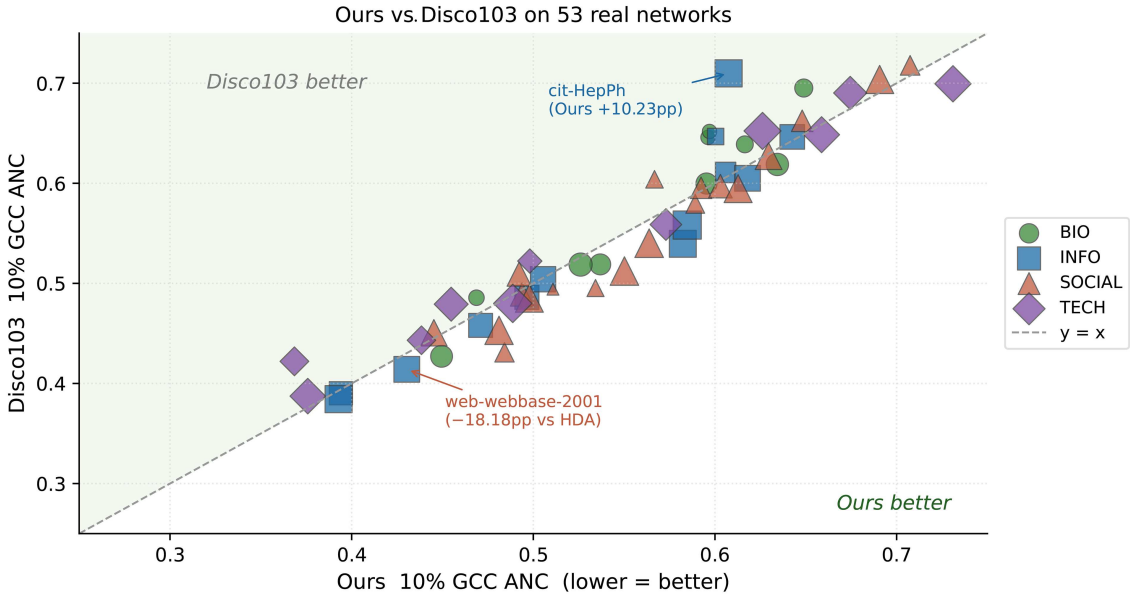

Figure 5 (Color online) Ours vs. Disco103 on the 53-network single-layer benchmark. Points below the diagonal favor Ours; colors indicate network categories.

Table 2 Zero-shot node-scale generalization on BA $(n, m = 4)$ graphs (mean ANC; column-best in bold, second-best underlined).

Method	$n = 10^2$	$n = 10^3$	$n = 10^4$
Ours	0.2420	0.2265	**0.2233**
FINDER	**0.2368**	**0.2254**	0.2238
Disco103	0.2427	0.2274	0.2243
CoreHD	0.2462	0.2281	0.2244
CI	0.2433	0.2292	0.2262
GND	0.2436	0.2638	0.3174
HDA	0.2467	0.2299	0.2259

6.2 Node-scale generalization

On synthetic BA graphs, Ours remains competitive as graph size increases from 10^2 to 10^4 (Table 2). FINDER has the lowest mean ANC at the two smaller sizes, while Ours has the lowest observed mean at $n = 10^4$. Ours outperforms Disco103 and all four classical baselines at every reported size.

Ours does not degrade with scale: its ANC is flat-to-improving ($0.242 \to 0.226 \to 0.223$) as n grows 200× past its training regime, while GND degrades sharply ($0.244 \to 0.264 \to 0.317$). At $n = 10^2$ only FINDER leads Ours; this gap is not statistically significant at $n = 10^3$, and Ours has the lowest observed mean at $n = 10^4$. These results are consistent with node-scale generalization of the meta-learned rule, though the small number of test graphs at $n = 10^4$ limits the strength of the conclusion.

Table 3 Mean ANC on 33 real-world multilayer networks under the interdependent cascade-failure protocol; Ours and Disco103 are zero-shot from single-layer-network dismantling agents. In the mean ANC column the lowest value is in bold and the second-lowest underlined.

Method	Mean ANC	vs. HDA	vs. Disco103	vs. Ours	vs. EMD	vs. MD
MD	**0.1296**	26/1/6	26/2/5	15/3/15	19/3/11	–
Ours	<u>0.1310</u>	25/1/7	25/3/5	–	16/6/11	15/3/15
EMD	0.1324	27/1/5	25/4/4	11/6/16	–	11/3/19
HDA	0.1523	–	23/2/8	7/1/25	5/1/27	6/1/26
Disco103	0.1775	8/2/23	–	5/3/25	4/4/25	5/2/26

Table 4 Mean ANC on 33 real-world multilayer networks under the aggregated multiplex protocol (Eq. (6)). The aggregated graph $\mathcal{A}$ is the edge-union of the two layers; dismantling is the standard 10%-threshold ANC on $\mathcal{A}$, with no inter-layer cascade. All six methods are evaluated zero-shot: none has seen a multiplex aggregated graph during training. In the Mean ANC column the lowest value is in bold and the second-lowest underlined.

Method	Mean ANC	vs. HDA	vs. FINDER	vs. EMD	vs. MD	vs. Ours
Ours	**0.2200**	**23/2/8**	**24/1/8**	**33/0/0**	**33/0/0**	–
Disco103	<u>0.2204</u>	23/1/9	26/1/6	32/0/1	32/0/1	16/3/14
HDA	0.2323	–	15/11/7	31/0/2	33/0/0	8/2/23
FINDER	0.2323	7/11/15	–	30/0/3	32/0/1	8/1/24
EMD	0.2598	2/0/31	3/0/30	–	29/0/4	0/0/33
MD	0.3203	0/0/33	1/0/32	4/0/29	–	0/0/33

6.3 Transfer to interdependent network dismantling

We evaluate the single-layer-trained agents zero-shot on 33 real-world multilayer networks under the interdependent cascade-failure protocol (Table 3).

Ours achieves mean ANC 0.1310, ranking second of five methods. It trails only the multilayer-specialized MD, with which it is tied head-to-head (15/3/15), and beats EMD, HDA, and Disco103. The networks where Ours beats MD and EMD are concentrated in the small-to-mid-size, cascade-light regime. Disco103, also zero-shot from single-layer training, ranks last with mean ANC 0.1775, suggesting that the graph-aligned inductive biases in our meta-network transfer more readily than those in the Atari-based Disco103 architecture; however, the precise mechanism behind this difference is not isolated by these experiments. Appendix O reports results on three larger SacchPomb genetic multilayer networks ($n_{\mathrm{per}} \in \{668, 1358, 2324\}$).

6.4 Transfer to multiplex network dismantling

We evaluate all methods zero-shot on the 33 multilayer networks under the aggregated multiplex protocol, where the target is the GACC of the edge-union graph $\mathcal{A}$ (Table 4).

Ours achieves the lowest mean ANC (0.2200), edging Disco103 (0.2204); both are tied head-to-head (16/3/14, sign-test $p = 0.86$). FINDER and HDA both score 0.2323, well behind the two meta-RL agents. EMD and MD, specialized for cascade-failure multilayer tasks, perform poorly here: the aggregated single-layer target does not match the objective they were optimized for.

FINDER (mean ANC 0.2323) does no better than training-free HDA (0.2323) and loses to Ours on 24 of 33 networks, indicating that our meta-learned update rule transfers across the single-layer-to-multiplex boundary more effectively than FINDER's hand-designed n-step Q-learning. Among the remaining methods, EMD is beaten on every network by Ours, and MD—the DRL SOTA on the cascade-failure measure—drops to last position.

6.5 Multi-task learning and cross-size transfer

The five-task meta-training distribution includes MIS, MVC, and MC. Table 5 reports results at the training scale $n \in [30, 50]$ and zero-shot at $n = 100$ and 200.

Ours comes within 0.4%–1.0% of the exact optimum on all three tasks at the training scale and beats the greedy baseline on every task and size. The MVC margin over greedy widens from -1.0% at the training scale to $-1.8\%/-2.0\%$ at $n = 100/200$, consistent with size generalization of the learned rule. Ours uniquely beats Disco103 on MVC; on MIS and MC the two tie under validation selection. Seed-to-seed stability results are in Appendix N.

Table 5 Cross-task generalization on MIS/MVC/MC across graph sizes. For each task, the frozen dismantling meta-network's task agents (best of 5 seeds, no retraining) are evaluated on 20 BA validation graphs at the training scale and, zero-shot, at $n = 100$ and 200. The exact Bron-Kerbosch optimum is shown where tractable; Δ is Ours' relative margin over the per-task greedy heuristic (negative = Ours better). Ours' best agent is in bold.

Task	Nodes	Greedy	Exact opt.	Ours best	Δ vs. greedy (%)
MIS	30–50	0.5947	0.5867	**0.5895**	−0.9
	100	0.5730	–	**0.5695**	−0.6
	200	0.5570	–	**0.5535**	−0.6
MVC	30–50	0.5984	0.5867	**0.5926**	−1.0
	100	0.5835	–	**0.5730**	**−1.8**
	200	0.5700	–	**0.5588**	**−2.0**
MC	30–50	0.8888	0.8793	**0.8825**	−0.7
	100	0.9590	0.9535	**0.9560**	−0.3
	200	0.9788	0.9765	**0.9770**	−0.2

Table 6 Ablation study. Mean ANC on the 53-network single-layer benchmark and 33-network zero-shot GMCC benchmark. Bold indicates the best result; underline the second-best.

Configuration	Multi-task	Meta	Graph	Single ANC ↓	GMCC ANC ↓
Ours	✓	✓	✓	**0.0878**	**0.1310**
Single-task	×	✓	✓	0.1419	0.1514
MTR	✓	×	✓	<u>0.0905</u>	0.1513
STR	×	×	✓	0.0937	<u>0.1357</u>
Disco103	✓	✓	×	0.0919	0.1775

6.6 Ablation study

We isolate the contributions of multi-task training, meta-learning, and graph-domain specialization. Table 6 compares five configurations on the 53-network single-layer and 33-network zero-shot GMCC benchmarks.

Removing multi-task training substantially degrades both benchmarks: single-task ANC rises from 0.0878 to 0.1419 on the single-layer benchmark and from 0.1310 to 0.1514 on GMCC.

Removing meta-learning while keeping multi-task training (MTR) preserves competitive single-layer ANC (0.0905) but degrades zero-shot GMCC transfer (0.1513). The single-task REINFORCE variant (STR) reaches 0.0937 single-layer and 0.1357 GMCC, showing that a hand-designed policy-gradient update can match Ours on single-layer while falling short on cross-protocol transfer.

Replacing the graph-domain meta-network with Disco103 raises single-layer ANC to 0.0919 and GMCC ANC to 0.1775; the larger gap on GMCC is consistent with the hypothesis that graph-domain specialization matters most when the test protocol differs from training.

Additional sensitivity analyses of alternative RL update rules, meta-network capacity, and the recurrent meta-state component are in Appendix M.

6.7 Policy-score diagnostics

We analyze whether the meta-network produces a task agent whose score distribution changes systematically around critical-node removals.

Setup. We evaluate 10 BA test graphs with $n \in \{30, 100, 500\}$ (511 steps total). At each step, the agent outputs a score vector over alive nodes; its standard deviation d_t measures how spread the scores are. Steps in the top 10% of GCC drops are *salient*; steps in the bottom 50% are *routine*. The detection statistic $D = \Pr(d_{\text{salient}} < d_{\text{routine}}) + \frac{1}{2}\Pr(d_{\text{salient}} = d_{\text{routine}})$ (Eq. (12)) measures how reliably salient steps have lower dispersion than routine ones; $D = 0.5$ is chance. We apply the same analysis to REINFORCE, PPO, and Disco103 controls trained under identical conditions.

Results. Figure 6 shows the per-step relationship between ΔGCC and score std for Ours (Pearson $r = -0.34$, $p < 10^{-15}$). Salient steps have lower score dispersion than routine steps on average (2.69×, median 3.94×). Ours reaches $D = 0.877$, while the best control (Disco103) reaches only $D = 0.741$; REINFORCE is anti-correlated ($D = 0.252$, sharper on routine steps) (Figure 7). This separation is specific to the graph-aligned discovered rule and is not a generic property of RL training.

This lower dispersion at salient steps means the agent concentrates score mass on fewer candidates when a critical node exists—a structural property of the scoring function, not a calibrated probability estimate. Whether

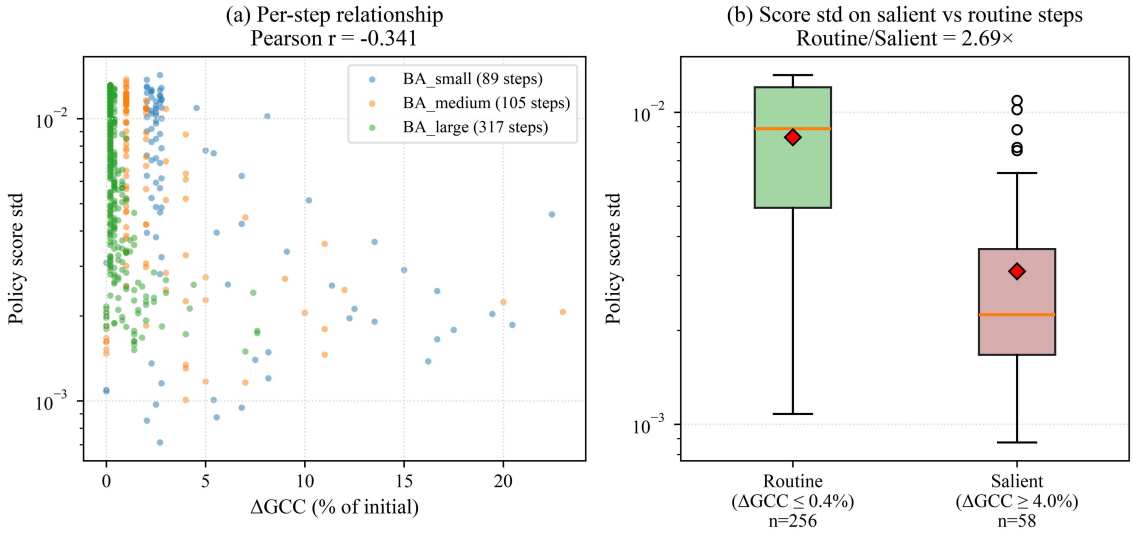

Figure 6 (Color online) Score dispersion at salient vs. routine steps. (a) Per-step relationship between ΔGCC (x-axis) and policy score std (y-axis, log scale) across 511 steps; (b) distribution of score std on routine vs. salient steps. Salient steps have lower dispersion on average.

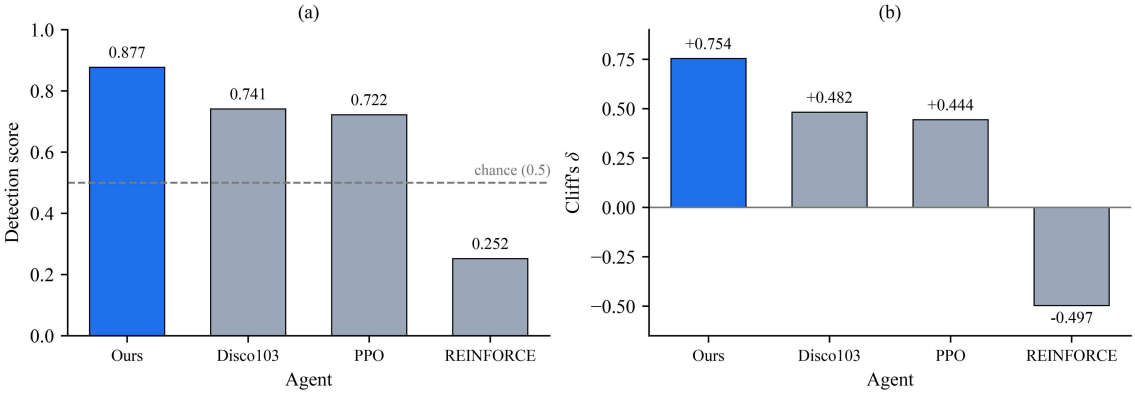

Figure 7 (Color online) Detection statistic and effect size for Ours and three control agents. (a) Detection score D: Ours ($D = 0.877$) substantially exceeds Disco103 ($D = 0.741$); REINFORCE is anti-correlated ($D = 0.252$). (b) Effect size (Cliff's δ), defined as $\Pr(d_{\text{salient}} < d_{\text{routine}}) - \Pr(d_{\text{salient}} > d_{\text{routine}})$; positive values indicate lower score dispersion on salient steps.

this reflects genuine recognition of critical nodes or a correlated feature of the learned distribution cannot be determined from these data alone.

Two complementary analyses in the supporting information quantify which structural properties drive the agent's scoring. A Kendall-τ correlation on 100 BA graphs shows the agent's score ranking agrees most strongly with degree ($\tau = 0.924$) and betweenness ($\tau = 0.835$), while correlating weakly with k-core ($\tau = 0.206$) (Appendix Q). Gradient saliency further shows that the agent is more sensitive to absent two-hop edges than to present ones (direct/two-hop gradient ratio 0.66), consistent with implicit evaluation of structural holes (Appendix R).

7 Discussion

7.1 Why a graph-aligned discovered rule wins

A graph-aligned discovered RL update rule outperforms the video-game-domain Disco103 baseline across all three evaluation regimes despite using a smaller meta-network and far smaller meta-training budget. The active factors are domain alignment of the discovered signal and of the architecture that carries it.

Distribution alignment. Disco103's meta-training on 103 video-game environments implies inductive biases (long episodes, sparse-reward exploration, grid-world navigation) not shared by graph optimization problems. Our 5-task graph distribution shares all relevant biases (short episodes, structure-driven action selection, sequential combinatorial optimization) with the evaluation tasks. As environment-inference work makes explicit [34, 35],

generalization is governed by whether the latent environment factors that drive meta-training are invariant across deployment, rather than by meta-training scale or capacity.

Architectural compression. Disco103's 600-dim prediction targets were chosen for Atari-scale return distributions. Graph episodes are 30–50 steps with bounded reward; 128-dim targets suffice with no performance loss, shrinking y/z head parameters by 9.3×.

Regularization via right-sizing. Disco103's 754k parameters appear overparameterized for graph problems, producing noisier loss landscapes that occasionally collapse; seed stability details are in Appendix N. The graph-sized configuration produces smooth training across all seeds; the ≈4.8× parameter saving is a by-product of right-sizing.

Scale-invariant generalization from small-graph training. The meta-network is trained only on small synthetic BA graphs yet generalizes to real networks up to $n = 79116$ in single-layer evaluation and $n_{\text{per}} = 2324$ in multilayer evaluation without performance degradation. Three architectural properties enable this. First, the 3-layer GNN's effective receptive field spans only 3-hop neighborhoods, ensuring decisions are based on local structural patterns; gradient saliency (Appendix R) confirms the 2-hop band is dominant (41%–61% across networks). Second, real networks widely exhibit scale-free degree distributions, so critical structural motifs in large graphs already exist in small BA graphs. Third, message-passing learns permutation-invariant local topological features rather than global statistics. Evaluation from 10^2 to 10^4 BA graphs shows stable ANC ($0.242 \to 0.226 \to 0.223$) with strengthening relative advantage as scale increases.

7.2 The dual-measure reversal: measure-specific vs. measure-robust SOTA

The clearest empirical signature of our meta-network is not that it beats MultiDismantler on average but that it transfers predictably across the two multiplex connectivity measures, whereas the multilayer-specialized baselines do not. Under GMCC, MD edges Ours (0.1296 vs. 0.1310 mean ANC, a head-to-head tie) by virtue of being trained on the exact multilayer mutual-connectivity reward signal that the connectivity measure scores; Ours trails by only 1% and ranks second, ahead of the multiplex heuristic EMD. Under the aggregated multiplex measure, the ordering reverses against the measure-specialized baselines: Ours attains a mean ANC 31% lower than MD's (0.2200 vs. 0.3203), beats the multiplex-specialized heuristic EMD on every network, and attains the lowest mean ANC overall, edging Disco103 (0.2204) at the top of the table.

We read this reversal as evidence about what each method has actually learned. MD's cascade-failure reward signal is geometrically distinct from the aggregated graph's GCC; EMD's iterative weight propagation encodes multiplex degree and inter-layer overlap, neither preserved by aggregation. By contrast, the aggregated graph is a single-layer graph, so our agent's single-layer training is a direct inductive match. The 5-task meta-training mix contributes the same shaping enabling zero-shot transfer in the opposite direction (single-to-cascade) at $\leqslant$7% cost. Measure-specialized training delivers in-measure wins but loses across measures, while a graph-aligned meta-network produces an agent that is the only one to remain non-dominated across all three protocols without retraining (single-layer, GMCC, GACC).

A practitioner-facing corollary: when the deployment metric is uncertain or expected to shift, the meta-network's measure-robust posture is preferable; when the metric is known and fixed, measure-specialized training retains the in-measure edge quantified below.

When and why Ours underperforms measure-specialized baselines. Two concrete conditions explain the observed gaps. (a) Measure proximity: the gap to a specialist grows when the evaluation measure is geometrically close to that specialist's training reward. MD leads Ours by only 1% on GMCC but trails by 31% on GACC, where its cascade reward is misaligned with the aggregated-graph objective. (b) Cascade dynamics: the advantage of GMCC-specialized training is concentrated on small, tightly coupled networks. On the three larger SacchPomb networks ($n_{\text{per}} = 668/1358/2324$) the per-network gap narrows from 26% to 3% as n_{per} grows, for a 9% mean (Appendix O). Prefer a cascade-specialized method only when the metric is cascade-failure-based, and networks are small and tightly coupled; prefer Ours otherwise.

7.3 Limitations

Measure-specialized baselines retain an in-measure edge. Under GMCC, MD leads Ours by 1% on the 33-network benchmark (Table 3) and by 9% on the three larger SacchPomb networks, where the per-network gap narrows from 26% to 3% as n_{per} grows from 668 to 2324 (Appendix O). Our advantage is cross-measure robustness rather than universal SOTA; extending meta-training to a multilayer task distribution would likely close this in-measure gap.

Synthetic small-graph training distribution and dense-adjacency sorter. All task agents are trained only on small synthetic BA graphs, and the current implementation materializes a $|V_{\text{merged}}|^2$ adjacency on GPU

at every removal step. Evaluation already reaches $n = 79116$ for single-layer and $n_{\text{per}} = 2324$ for multilayer (Appendix O), so the headline results are not graph-size-limited; pushing to $n_{\text{per}} \geqslant 10^4$ multilayer networks (e.g., Sanremo2016, NYClimate2014) and to real-world network meta-training mixtures will require a sparse-adjacency sorter, which we view as direct future work rather than a fundamental limitation. Recent continuous-relaxation dismantling [20] and contrastive-representation learning [21] both demonstrate that decoupling the removal-decision module from a dense adjacency materialization is achievable without loss of dismantling quality; combining that decoupling with a discovered update rule is the natural next step.

Evaluation scale of MIS/MVC/MC. The cross-task results use BA $n = 30$–50 so Bron-Kerbosch exact optima are tractable; in this regime greedy is already within 1%–2% of the optimum, which bounds the visible margin. Carrying the comparison to DIMACS-clique-scale benchmarks ($n = 200$–4000) is left to future work.

Truncated meta-gradient backpropagation. Following Oh et al. [6], we unroll $M = 2$ differentiable inner steps for meta-gradient computation, backpropagating only through the last M steps rather than the full $K=$ 200-step inner roll-out. This truncation trades gradient fidelity for computational tractability. Exploring larger M or unbiased meta-gradient estimators is a natural direction for future work on richer task distributions.

8 Conclusion

We discover a compact graph-domain meta-RL rule, meta-trained on five graph-combinatorial tasks on a single A800 GPU in under one day, whose task agents empirically dominate across single-layer, multilayer, and graph-combinatorial benchmarks. On the established 53-network single-layer network dismantling benchmark the agents outperform all baselines. On 33 real-world multilayer networks of both types, they transfer zero-shot, approaching the specialized SOTA (MD) on interdependent networks and surpassing all baselines on multiplex networks. On MIS, MVC, and MC they come within ≈1% of the exact optimum without per-task tuning.

A prediction-semantics analysis shows that the meta-network produces a task agent whose score dispersion is lower on salient (critical-node) steps than on routine steps (dispersion ratio ≈2.7×, detection statistic $D = 0.877$ vs. $D = 0.741$ for the best control). These empirical advantages are supported by a strict fair-protocol ablation in which the agent backbone, training distribution, random seeds, inner-step budget, and loss-coefficient defaults are held identical across update rules; under this protocol, the discovered, graph-domain-specific RL signal itself is the load-bearing factor and is not reproduced by classical REINFORCE or PPO, by a smaller meta-network variant ($p = 64$), or by Disco103's larger capacity.

Outlook. Adding a multilayer cascade-failure task to the meta-training mixture should close the residual gap to MD on its own protocol. Expanding the meta-training distribution to include Erdős-Rényi topologies, heterogeneous cost structures (time-complexity vs. attack-cost tradeoffs), and scale-stratified training regimes could produce agents that trade off dismantling effectiveness for execution speed under deployment constraints. Beyond network dismantling, the graph-domain meta-RL framework applies to further network-science tasks: hypergraph problems, network monitoring, and sensor-placement optimization. On the engineering side, the current agent materializes a dense $|V|^2$ adjacency and re-encodes it through the GNN at every removal step, which becomes the wall-clock bottleneck once n_{per} approaches a few thousand; a sparse-adjacency sorter that updates incrementally between steps would extend zero-shot evaluation to larger multilayer networks; cross-task generalization to larger combinatorial benchmarks is similarly left to future work. More broadly, the work demonstrates that for complex-network problems, compact, domain-aligned meta-learning dominates large, domain-agnostic meta-learning, and that the source of this advantage is the discovered graph-domain RL signal itself rather than any single component of the underlying neural architecture.

Supporting information Appendixes A–U. The supporting information is available online at info.scichina.com and link.springer.com. The supporting materials are published as submitted, without typesetting or editing. The responsibility for scientific accuracy and content remains entirely with the authors.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant No. 62572480), Major Scientific and Technological Innovation Platform Project of Hunan Province (Grant No. 2024JC1003), and National Natural Science Foundation of China (Grant No. 72401290).

References

1 Morone F, Makse H A. Influence maximization in complex networks through optimal percolation. Nature, 2015, 524: 65–68
2 Buldyrev S V, Parshani R, Paul G, et al. Catastrophic cascade of failures in interdependent networks. Nature, 2010, 464: 1025–1028
3 Braunstein A, Dall'Asta L, Semerjian G, et al. Network dismantling. Proc Natl Acad Sci USA, 2016, 113: 12368–12373
4 Fan C, Zeng L, Sun Y, et al. Finding key players in complex networks through deep reinforcement learning. Nat Mach Intell, 2020, 2: 317–324
5 Williams R J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach Learn, 1992, 8: 229–256
6 Oh J, Farquhar G, Kemaev I, et al. Discovering state-of-the-art reinforcement learning algorithms. Nature, 2025, 648: 312–319

7 Finn C, Abbeel P, Levine S. Model-agnostic meta-learning for fast adaptation of deep networks. In: Proceedings of the International Conference on Machine Learning, 2017. 1126–1135
8 Andrychowicz M, Denil M, Gomez S, et al. Learning to learn by gradient descent by gradient descent. In: Proceedings of the Conference on Neural Information Processing Systems, 2016. 3981–3989
9 Artime O, Grassia M, De Domenico M, et al. Robustness and resilience of complex networks. Nat Rev Phys, 2024, 6: 114–131
10 Zdeborová L, Zhang P, Zhou H J. Fast and simple decycling and dismantling of networks. Sci Rep, 2016, 6: 37954
11 Schneider C M, Moreira A A, Andrade Jr. J S, et al. Mitigation of malicious attacks on networks. Proc Natl Acad Sci USA, 2011, 108: 3838–3841
12 Albert R, Jeong H, Barabási A L. Error and attack tolerance of complex networks. Nature, 2000, 406: 378–382
13 Cohen R, Erez K, ben-Avraham D, et al. Breakdown of the internet under intentional attack. Phys Rev Lett, 2001, 86: 3682–3685
14 Holme P, Kim B J, Yoon C N, et al. Attack vulnerability of complex networks. Phys Rev E, 2002, 65: 056109
15 Kitsak M, Gallos L K, Havlin S, et al. Identification of influential spreaders in complex networks. Nat Phys, 2010, 6: 888–893
16 Lü L, Chen D, Ren X L, et al. Vital nodes identification in complex networks. Phys Rep, 2016, 650: 1–63
17 Deng Y, Wang Z, Xiao Y, et al. Spatial network disintegration based on spatial coverage. Reliability Eng Syst Saf, 2025, 253: 110525
18 Wu M, Mou J, Dai B, et al. Dismantling directed networks: a multi-temporal information field approach. Chaos Solitons Fractals, 2025, 196: 116404
19 Tian H, Ferraro P, Shorten R, et al. Learning network dismantling without handcrafted inputs. 2025. ArXiv:2508.00706
20 Liu Y, Chen X, Wang X, et al. Efficient continuous network dismantling. IEEE Trans Syst Man Cybern Syst, 2025, 55: 976–989
21 Ma S Q, Zeng W X, Xiao W D, et al. Dismantling complex networks with graph contrastive learning and multi-hop aggregation. Inf Sci, 2024, 676: 120780
22 Zhang P, Fu J, Sun X J, et al. Neural algorithmic approach to network dismantling. Sci China Inf Sci, 2026, 69: 119203
23 Kivelä M, Arenas A, Barthelemy M, et al. Multilayer networks. J Complex Netw, 2014, 2: 203–271
24 Boccaletti S, Bianconi G, Criado R, et al. The structure and dynamics of multilayer networks. Phys Rep, 2014, 544: 1–122
25 Osat S, Faqeeh A, Radicchi F. Optimal percolation on multiplex networks. Nat Commun, 2017, 8: 1540
26 Hamilton W L, Ying R, Leskovec J. Inductive representation learning on large graphs. In: Proceedings of the Conference on Neural Information Processing Systems, 2017. 1025–1035
27 Veličković P, Cucurull G, Casanova A, et al. Graph attention networks. In: Proceedings of the International Conference on Learning Representations, 2018
28 Cai W Y, Li M F, Shi X H, et al. RE-SEGNN: recurrent semantic evidence-aware graph neural network for temporal knowledge graph forecasting. Sci China Inf Sci, 2025, 68: 122104
29 Dai H, Khalil E B, Zhang Y, et al. Learning combinatorial optimization algorithms over graphs. In: Proceedings of the Conference on Neural Information Processing Systems, 2017. 6348–6358
30 Gu W, Yang C, Li L, et al. Deep-learning-aided dismantling of interdependent networks. Nat Mach Intell, 2025, 7: 1266–1277
31 Sutton R S, Barto A G. Reinforcement Learning: an Introduction. 2nd ed. Cambridge: MIT Press, 2018
32 Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning. Nature, 2015, 518: 529–533
33 Schulman J, Wolski F, Dhariwal P, et al. Proximal policy optimization algorithms. 2017. ArXiv:1707.06347
34 Wang H, Kuang K, Lan L, et al. Out-of-distribution generalization with causal feature separation. IEEE Trans Knowl Data Eng, 2024, 36: 1758–1772
35 Liu S, He Y, Wang H, et al. Environment inference for learning generalizable dynamical system. In: Proceedings of the Conference on Neural Information Processing Systems, 2025. 164076–164105