SCIENCE CHINA
Information Sciences

• Supplementary File •

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*

Appendix A Meta-network architecture: per-component parameter breakdown

The graph-domain meta-network described in Section 4.1 of the main paper has 155K parameters distributed across six components. Table A1 shows the breakdown alongside the video-game-domain reference Disco103 (754K) [2].

Table A1 Per-component parameter comparison between our 155K graph-domain meta-net and DeepMind's Disco103 (754K). The largest single saving comes from the output projections ($y_{\text{head}} + z_{\text{head}}$, 9.3×), driven by reducing the prediction-target dimension from 600 to 128 — appropriate for the short, bounded-reward graph episodes of 30–50 steps used here.

Component	Ours (ps=128)	Disco103 (ps=600)	Ratio
y_net + z_net (input embedding)	4162	19266	4.6×
trajectory_rnn (LSTM hidden 128 vs 256)	79872	290816	3.6×
state_gate (multiplicative interaction)	8320	33024	4.0×
y_head + z_head (output projections)	33024	308400	**9.3**×
meta_rnn_cell (lifetime LSTM 64 vs 128)	20736	74240	3.6×
remainder (policy / conv / MLPs)	9704	29032	3.0×
Total	**155818**	**754778**	**4.84**×

Appendix B Multi-task mix as the source of cross-protocol transfer

The multi-task mixture produces transferable agents on out-of-distribution multilayer benchmarks (Sections 6.3 and 6.4 of the main paper), whereas a single-task agent trained directly on the multilayer reward signal worsens to mean AUC 0.2327 on the 33-network cascade-failure benchmark (Section 7.2 of the main paper). The interpretation is that the useful cascade-aware update dynamics emerge from the multi-task mix — single-layer dismantling alongside the four other tasks — rather than from per-task fresh-agent training.

Appendix C Complete hyper-parameter listing

Table C1 collects every hyper-parameter used in this paper. None are tuned per evaluation network or per meta-net; the same settings produce all reported numbers.

Appendix D Per-network single-layer AUC, 53 networks

Table D1 reports the per-network 10%-LCC-threshold AUC for the 53 networks of Artime et al. [1] on which the headline result of Section 6.1 of the main paper is computed. Bold marks the per-row minimum across the four methods.

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

Table C1 Hyper-parameter values used throughout the paper.

Group	Parameter	Value
Meta-training	meta-train tasks	5 (single, ml-gmcc, MIS, MVC, MC)
	train graphs	BA, $n \in [30, 50]$, $m = 4$
	outer steps	200
	inner steps per outer	$K = 200$
	differentiable inner steps	$M = 2$
	outer batch (tasks × agents)	32
Meta-optimisation	optimiser	Adam
	learning rate	3×10^{-4}
	gradient-norm clip	0.5
	weight decay	0
Meta-net	prediction dim p	128
	trajectory LSTM hidden	128
	meta-RNN cell hidden	64
	total parameters	155818
Fresh agent (GNN)	layers	3 message-passing rounds
	embed / hidden dim	64 / 128
	node features	alive + degree + clust. coeff. + k-shell + eigvec. centr.
	total parameters	$\sim$ 155000
	fresh-agent optimiser	Adam, lr 10^{-2}
	fresh-agent inner steps (eval)	500
Inner-loop loss	α_π, α_y, α_z, α_{aux}	1.0 (all four)
	α_v	0.2
	λ_v (value-bootstrap mix)	0.95
	κ (auxiliary-policy temp.)	0.9
	policy-gradient baseline b_k	$(R_k^{\text{HDA}} + R_k^{\text{rand}})/2$, fixed per task
	val. baseline graphs per task	128
Evaluation protocol	single-layer threshold	10%-LCC
	batch-removal step	$\max(1, \lceil 0.01n \rceil)$
	agent-selection set	BA validation set (disjoint from 53 test)
	agent-selection rule	best val-AUC of 10 seeds
REINFORCE abl.	lr	10^{-2}
	baseline	mean return
	inner steps	500
PPO ablation	lr	3×10^{-4}
	clip ϵ	0.2
	value-coeff	0.05 (normalised returns)
	inner epochs K	4

Table D1 Per-network 10%-LCC-threshold AUC on the 53 single-layer networks of Artime et al. [1]. N_0 is the initial node count. Lower is better; bold marks the row best.

Network	Cat.	N_0	**Ours**	Disco103	HDA	FINDER
arenas-meta	BIO	453	0.0860	0.0922	**0.0845**	0.1582
dimacs10-celegansneural	BIO	297	0.2653	**0.2482**	0.2681	0.2709
foodweb-baydry	BIO	128	0.3764	**0.3558**	0.3791	0.3772
foodweb-baywet	BIO	128	0.3732	**0.3543**	0.3765	0.3727
HI-II-14	BIO	4165	**0.0535**	0.0575	0.0553	0.0574
maayan-figeys	BIO	2217	**0.0296**	0.0355	0.0301	0.0351
maayan-foodweb	BIO	183	**0.1695**	0.1737	0.1935	0.2323
maayan-Stelzl	BIO	1615	**0.0407**	0.0485	0.0419	0.0484
maayan-vidal	BIO	2783	**0.0699**	0.0709	0.0717	0.0746
moreno_propro	BIO	1458	**0.0371**	0.0388	0.0378	0.0470

(continued on next page)

(Table D1 continued from previous page)

Network	Cat.	N_0	**Ours**	Disco103	HDA	FINDER
cfinder-google	INFO	15763	0.0407	0.1003	**0.0228**	0.1129
cit-HepPh	INFO	34401	0.3025	**0.2645**	0.3074	0.2991
corruption	INFO	309	**0.0433**	0.0871	0.0488	0.1940
dblp-cite	INFO	12495	0.0954	**0.0889**	0.1056	0.1127
dimacs10-polblogs	INFO	1222	0.1820	**0.1813**	0.1919	0.1951
econ-wm1	INFO	258	**0.1663**	0.1692	0.1702	0.2122
internet-topology	INFO	34761	0.0139	0.0233	**0.0125**	0.0194
moreno_train_train	INFO	64	0.1264	0.1329	**0.1245**	0.1614
subelj_jdk_jdk	INFO	6434	**0.0368**	0.0507	0.0421	0.3681
subelj_jung-j_jung-j	INFO	6120	**0.0366**	0.0479	0.0421	0.3782
web-EPA	INFO	4253	**0.0329**	0.0336	0.0339	0.0664
web-webbase-2001	INFO	16062	0.0042	0.0065	**0.0010**	0.0088
advogato	SOCIAL	5042	**0.1395**	0.1410	0.1432	0.1452
Crime	SOCIAL	829	0.1050	0.1043	0.1062	**0.0679**
ego-twitter	SOCIAL	22322	0.0093	0.0116	**0.0087**	0.0128
Facebook	SOCIAL	63392	0.2649	**0.2336**	0.2714	0.2657
librec-ciaodvd-trust	SOCIAL	4562	0.0785	**0.0766**	0.0944	0.0991
librec-filmtrust-trust	SOCIAL	610	0.0540	**0.0534**	0.0557	0.0637
london_transport_multiplex_aggr	SOCIAL	369	**0.0438**	0.0457	0.0450	0.0470
moreno_crime_projected	SOCIAL	754	**0.0183**	0.0276	0.0235	0.0341
munmun_digg_reply_LCC	SOCIAL	29652	**0.0855**	0.0884	0.0868	0.0890
opsahl-ucsocial	SOCIAL	1893	0.1402	**0.1394**	0.1443	0.1430
pajek-erdos	SOCIAL	6927	0.0194	0.0260	**0.0179**	0.0225
petster-cat-household	SOCIAL	68315	**0.0416**	0.0518	0.0424	0.0435
petster-hamster	SOCIAL	2000	0.1634	**0.1517**	0.1695	0.1941
slashdot-threads	SOCIAL	51083	**0.0324**	0.0381	0.0326	0.0329
slashdot-zoo	SOCIAL	79116	**0.0673**	0.0806	0.0688	0.0714
soc-Epinions1	SOCIAL	75877	**0.0485**	0.0665	0.0505	0.0536
Enron	TECH	33696	**0.0377**	0.0444	0.0408	0.0450
eu-powergrid	TECH	1093	0.0226	0.0279	**0.0226**	0.0357
gridkit-eupowergrid	TECH	13478	**0.0198**	0.0203	0.0200	0.0267
gridkit-north_america	TECH	14990	0.0249	**0.0236**	0.0252	0.0291
inf-USAir97	TECH	332	**0.0830**	0.0927	0.0939	0.0926
linux	TECH	30817	**0.0480**	0.0795	0.0500	0.0662
loc-brightkite	TECH	56739	**0.0820**	0.0863	0.0849	0.0886
opsahl-openflights	TECH	2905	**0.0651**	0.0717	0.0771	0.0856
opsahl-powergrid	TECH	4941	0.0402	**0.0399**	0.0420	0.0484
oregon2_010526	TECH	11461	**0.0148**	0.0213	0.0163	0.0233
p2p-Gnutella06	TECH	8717	0.1889	**0.1760**	0.1960	0.1858
p2p-Gnutella31	TECH	62561	0.1088	**0.1037**	0.1137	0.1199
power-eris1176	TECH	1174	0.0319	0.0549	**0.0218**	0.0996
roads-california	TECH	21048	0.0109	0.0110	**0.0107**	0.0126
route-views	TECH	6474	0.0122	0.0219	**0.0114**	0.0124

(continued on next page)

(Table D1 continued from previous page)

Network	Cat.	N_0	**Ours**	Disco103	HDA	FINDER
Mean			**0.0884**	0.0919	0.0911	0.1162

Appendix E Per-network multilayer interdependent (GMCC) AUC, 33 networks

Table E1 reports the per-network 10%-GMCC-threshold AUC for the 33 multilayer networks on which Section 6.3 of the main paper reports zero-shot transfer.

Table E1 Per-network 10%-GMCC-threshold AUC for 33 real-world multilayer networks under the cascade-failure protocol. n_{per} is the per-layer node count. Lower is better; bold marks the row best.

Network	n_{per}	**Ours**	Disco103	MD	EMD	HDA
CS-Aarhus_multiplex_edges_2_3	13	**0.0434**	**0.0434**	**0.0434**	**0.0434**	0.0493
Krackhardt-High-Tech_multiplex_edges_1_2	21	0.2971	0.3314	**0.2880**	0.2980	0.3836
Krackhardt-High-Tech_multiplex_edges_1_3	21	**0.0416**	0.0444	**0.0416**	0.0444	0.2094
Krackhardt-High-Tech_multiplex_edges_2_3	21	**0.0234**	**0.0234**	0.0385	**0.0234**	0.1164
CS-Aarhus_multiplex_edges_1_3	25	0.0483	0.0361	**0.0329**	0.0491	0.0544
EA_filtered2_32	25	**0.0016**	0.0048	**0.0016**	**0.0016**	0.0048
Vickers-Chan-7thGraders_multiplex_edges_1_2	29	**0.2533**	0.3995	0.2794	0.3000	0.2730
Vickers-Chan-7thGraders_multiplex_edges_1_3	29	0.2937	0.4296	**0.2889**	0.2953	0.3296
Vickers-Chan-7thGraders_multiplex_edges_2_3	29	**0.2747**	0.4474	0.2874	0.2885	0.3024
CS-Aarhus_multiplex_edges_1_2	32	0.1742	0.2132	**0.1348**	0.1437	0.1632
gallus_genetic_multiplex	37	**0.0140**	0.0263	0.0145	**0.0140**	**0.0140**
Kapferer-Tailor-Shop_multiplex-12	39	0.2632	0.3307	0.2715	**0.2623**	0.2831
EA_filtered1_6	45	**0.0367**	0.0391	0.0433	0.0398	0.0391
EA_filtered1_3	51	0.0145	0.0145	**0.0141**	0.0145	**0.0141**
Lazega-Law-Firm_multiplex_edges_1_2	69	**0.3290**	0.4560	0.3301	0.3485	0.3807
Lazega-Law-Firm_multiplex_edges_2_3	69	0.3459	0.4297	**0.3091**	0.3533	0.3663
Lazega-Law-Firm_multiplex_edges_1_3	71	**0.3773**	0.4760	0.3953	0.4022	0.4264
rattus_genetic_multiplex_edges_2_3	71	0.0067	0.0078	**0.0058**	**0.0058**	0.0063
US_real2_2	72	**0.0550**	0.0647	0.0558	0.0552	0.0629
US_real2_3	81	0.0447	0.0444	0.0461	**0.0442**	0.0667
US_real2_1	84	**0.0490**	0.0494	0.0541	0.0503	0.0738
rattus_genetic_multiplex_edges_1_3	106	**0.0019**	0.0084	0.0056	**0.0019**	0.0022
humanHIV1_genetic_multiplex-12	144	0.0059	**0.0054**	**0.0054**	0.0064	0.0057
fao_trade_multiplex-3-24	193	0.2902	0.4888	**0.2742**	0.2826	0.2915
CKM-Physicians-Innovation_multiplex_edges_1_3	205	0.0645	0.0738	0.0886	**0.0592**	0.0689
CKM-Physicians-Innovation_multiplex_edges_1_2	212	**0.0643**	0.0876	0.0654	0.0689	0.0656
CKM-Physicians-Innovation_multiplex_edges_2_3	218	0.1001	0.0976	0.0864	**0.0740**	0.0792
celegans_connectome_multiplex_edges_1_2	238	0.1197	0.1597	**0.1065**	0.1069	0.1236
celegans_connectome_multiplex_edges_1_3	252	0.1101	0.1615	**0.1084**	0.1109	0.1511
celegans_connectome_multiplex_edges_2_3	259	0.2027	0.2860	**0.1805**	0.1838	0.2116
rattus_genetic_multiplex_edges_1_2	454	0.0038	0.0034	0.0026	**0.0025**	0.0028
drosophila_melanogaster_multiplex-5-6	509	0.0721	0.1006	**0.0600**	0.0603	0.0682
fb-tw-12	1043	**0.3014**	0.4713	0.3182	0.3346	0.3347
Mean		0.1310	0.1775	**0.1296**	0.1324	0.1523

(continued on next page)

(Table E1 continued from previous page)

Network	n_{per}	**Ours**	Disco103	MD	EMD	HDA

Appendix F Per-network multilayer multiplex (GACC) AUC, 33 networks

Table F1 reports the per-network 10%-GACC-threshold AUC for the 33 networks under the multiplex (aggregated-graph) protocol of Section 6.4 of the main paper.

Table F1 Per-network 10%-GACC-threshold AUC for 33 real-world multilayer networks under the multiplex aggregated-graph protocol. Lower is better; bold marks the row best.

Network	n_{per}	**Ours**	Disco103	HDA	FINDER	EMD	MD
CS-Aarhus_multiplex_edges_2_3	13	**0.2032**	**0.2032**	0.2051	0.2051	0.2298	0.3570
Krackhardt-High-Tech_multiplex_edges_1_2	21	0.4059	**0.3893**	0.4104	0.4104	0.4182	0.4346
Krackhardt-High-Tech_multiplex_edges_1_3	21	0.3681	**0.3651**	0.3741	0.3741	0.3740	0.4433
Krackhardt-High-Tech_multiplex_edges_2_3	21	0.3061	**0.2986**	0.3243	0.3243	0.3413	0.4267
CS-Aarhus_multiplex_edges_1_3	25	0.0943	0.0976	0.0927	**0.0767**	0.1579	0.2132
EA_filtered2_32	25	**0.1957**	0.1979	0.2043	0.2043	0.1963	0.3792
Vickers-Chan-7thGraders_multiplex_edges_1_2	29	0.3607	**0.3567**	0.4082	0.4068	0.4257	0.4489
Vickers-Chan-7thGraders_multiplex_edges_1_3	29	0.3591	**0.3559**	0.4106	0.4114	0.4303	0.4440
Vickers-Chan-7thGraders_multiplex_edges_2_3	29	**0.2858**	0.3214	0.3207	0.3296	0.3650	0.3321
CS-Aarhus_multiplex_edges_1_2	32	0.3057	**0.2871**	0.3012	0.3012	0.3324	0.3821
gallus_genetic_multiplex	37	**0.0037**	0.0083	**0.0037**	0.0106	0.0388	0.0044
Kapferer-Tailor-Shop_multiplex-12	39	**0.3682**	0.3703	0.3751	0.3754	0.3936	0.4496
EA_filtered1_6	45	**0.1198**	**0.1198**	0.1205	0.1205	0.1212	0.2616
EA_filtered1_3	51	**0.0864**	0.0874	0.0887	0.0887	0.0946	0.2098
Lazega-Law-Firm_multiplex_edges_1_2	69	0.4153	**0.4083**	0.4329	0.4410	0.4377	0.4533
Lazega-Law-Firm_multiplex_edges_2_3	69	0.4167	**0.4047**	0.4214	0.4282	0.4385	0.4608
Lazega-Law-Firm_multiplex_edges_1_3	71	0.4186	**0.4167**	0.4457	0.4447	0.4500	0.4714
rattus_genetic_multiplex_edges_2_3	71	**0.0000**	**0.0000**	**0.0000**	**0.0000**	0.0284	0.0480
US_real2_2	72	0.1178	0.1233	**0.1154**	0.1174	0.1424	0.3351
US_real2_3	81	0.1728	0.1739	**0.1719**	0.1721	0.1954	0.4661
US_real2_1	84	0.1924	**0.1910**	0.2012	0.2033	0.2154	0.4013
rattus_genetic_multiplex_edges_1_3	106	0.0036	0.0046	**0.0030**	**0.0030**	0.0325	0.0635
humanHIV1_genetic_multiplex-12	144	0.0206	0.0204	**0.0192**	**0.0192**	0.0282	0.0574
fao_trade_multiplex-3-24	193	0.3400	**0.3279**	0.3712	0.3731	0.3862	0.4079
CKM-Physicians-Innovation_multiplex_edges_1_3	205	**0.1176**	0.1194	0.1300	0.1317	0.1700	0.2057
CKM-Physicians-Innovation_multiplex_edges_1_2	212	0.1028	**0.1026**	0.1093	0.1117	0.1691	0.1579
CKM-Physicians-Innovation_multiplex_edges_2_3	218	0.1481	**0.1421**	0.1425	0.1449	0.1694	0.2032
celegans_connectome_multiplex_edges_1_2	238	**0.2597**	0.2856	0.2794	0.2725	0.3288	0.4424
celegans_connectome_multiplex_edges_1_3	252	**0.2884**	0.3112	0.3327	0.3260	0.3811	0.4411
celegans_connectome_multiplex_edges_2_3	259	0.2786	**0.2785**	0.3099	0.3011	0.3495	0.3745
rattus_genetic_multiplex_edges_1_2	454	0.0482	0.0442	**0.0437**	0.0457	0.1428	0.1732
drosophila_melanogaster_multiplex-5-6	509	**0.1492**	0.1496	0.1592	0.1529	0.2324	0.2809
fb-tw-12	1043	**0.3059**	0.3106	0.3375	0.3378	0.3553	0.3400
Mean		**0.2200**	0.2204	0.2323	0.2323	0.2598	0.3203

Appendix G Meta-training trajectories

We record the meta-training trajectory of the 155K meta-net used throughout the main paper. Figure G1 shows the outer-loop loss $J(\theta_T)$ and the meta-gradient norm $\|\nabla_\varphi J\|$ across all 200 outer steps. Figure G2 shows the per-task validation AUC (lower is better) measured on the validation graphs at the end of every outer step, with the fixed REINFORCE baseline $b_k = (R_k^{\mathrm{HDA}} + R_k^{\mathrm{rand}})/2$ overlaid as a dashed horizontal line.

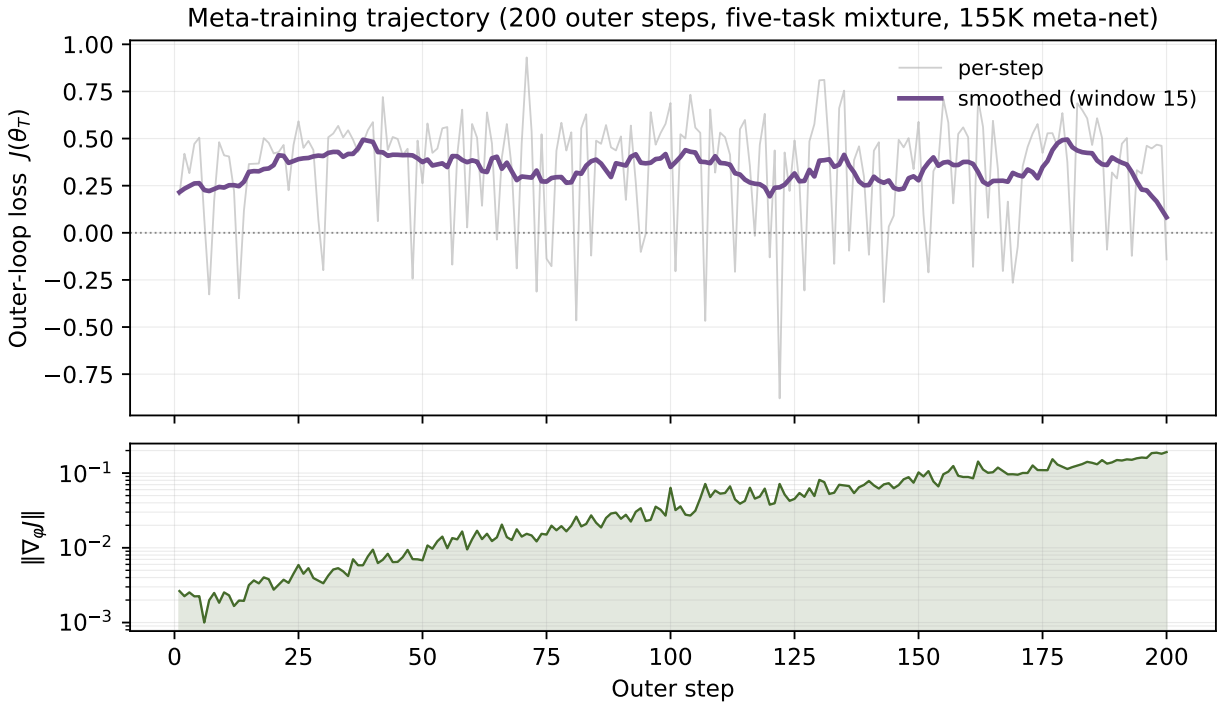

Figure G1 (Top) outer-loop loss $J(\theta_T)$ vs. outer step, faint grey is the per-step value and purple is a length-15 causal moving average. (Bottom) meta-gradient norm $\|\nabla_\varphi J\|$ on a log scale — it grows monotonically from $\sim 10^{-3}$ to $\sim 10^{-1}$ over the 200 outer steps, consistent with the meta-network gradually producing more informative loss-shaping targets as training progresses.

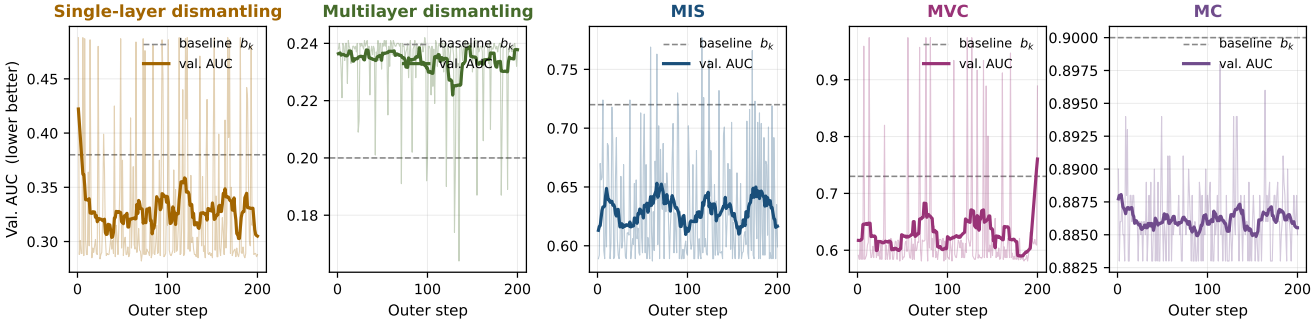

Figure G2 Per-task validation AUC over 200 outer meta-training steps (lower is better). Faint colour: raw per-step value; solid: length-15 smoothed; dashed grey: fixed baseline $b_k = (R_k^{\mathrm{HDA}} + R_k^{\mathrm{rand}})/2$ used as the REINFORCE control variate. All four non-saturated tasks (single-layer network dismantling, multilayer network dismantling, MIS, MVC) settle below their respective baselines after the first $\sim$30 outer steps and then drift down slightly; MC is already saturated near $b_{\mathrm{MC}} \approx 0.89$ because the exact-optimum gap on BA $n \in [30, 50]$ graphs is below 1% even for greedy. This is the trajectory underlying the headline results of Sections 6.1–6.6 of the main paper.

Appendix H Implementation notes

Software stack. PyTorch 2.4, NetworkX 3.4, graph-tool 2.77, Python 3.10. Disco103 weights are loaded from the official `disco_103.npz` release and converted to PyTorch tensors.

Random seeds. Fresh-agent training uses the seed sweep $\{1000, 1077, 1154, \ldots, 1693\}$ (ten evenly-spaced seeds with 77-step interval). Evaluation graphs use seed 42 for the synthetic BA validation graphs in Section 6.6 of the main paper.

Validation-graph sampler. The five per-task validation graph sets are drawn once at the start of meta-training and held fixed across the entire run. Single-layer and MIS/MVC/MC use 128 BA val graphs each at $n \in [30, 50]$; multilayer GMCC uses 128 interdependent

two-layer BA pairs at $n_{\mathrm{per}} \in [10, 25]$ (smaller because cascade simulation is the inner-loop bottleneck).

Dense-adjacency sorter. The current GNN re-encodes the full $|V_{\mathrm{merged}}|^2$ adjacency at every removal step, which is the wall-clock bottleneck on $n_{\mathrm{per}} \gtrsim 10^4$ multilayer networks. A sparse-adjacency incremental sorter would unblock this regime without changing the meta-net itself.

Interdependent network protocol (GMCC): full specification. The two-layer interdependent cascade is defined as follows (implemented in `envs/multilayer_dismantle_env.py`, `gmcc_size_and_cascade`). *(i) Coupling* is *one-to-one*: the two layers share one node-ID space, and the two replicas (i, i) of a node form its interdependency pair, so removing one replica removes its paired replica in the other layer. *(ii) Pruning rule.* Starting from the current removal set, we iterate: any surviving node that falls into a connected component of size $\leqslant 1$ (i.e. becomes isolated) in *either* layer is marked cascade-failed and removed from *both* layers; failures therefore propagate across layers. *(iii) Convergence and iteration cap.* The iteration runs to a fixed point — it stops at the first round that produces no new failed node. No artificial maximum-iteration cap is needed: every non-terminal round removes at least one node, so the cascade converges in at most $O(n_{\mathrm{per}})$ rounds. *(iv) GMCC.* After convergence, the giant mutually connected component is the largest intersection of a layer-1 connected component with a layer-2 connected component (found by BFS). The dismantling objective runs this full cascade after *every* removal step and checks the 10%-GMCC stopping threshold at the end of cascade convergence, not at the end of the removal step, matching the convention of Gu et al. [3].

Bron–Kerbosch exact optimum. For MIS / MVC / MC, the exact optimum is computed by NetworkX's `find_cliques_recursive` implementation of Bron–Kerbosch on graphs of $n \leqslant 50$; running time is sub-second per graph at this scale.

Fair-protocol concession to PPO. Section 6.7 of the main paper uses PPO's published default learning rate (3×10^{-4}) rather than the 10^{-2} used by REINFORCE and the meta-net; at lr 10^{-2} PPO's clipped surrogate is degenerate within the first few inner steps. All other PPO hyper-parameters follow the original paper [7].

Appendix I Baseline methods

This appendix details how each non-RL baseline ranks and removes nodes; the main text gives only the acronym expansions.

HDA (High-Degree Adaptive). HDA is the adaptive variant of the static High-Degree (HD) rule, which ranks nodes once by their initial degree [4]. HDA instead recomputes the degree of every remaining node after each removal and, at each step, removes the current highest-degree node. The "adaptive" qualifier refers to this per-step recomputation, which lets the attack track the topology as it fragments.

HDA in the interdependent protocol. In the two-layer interdependent setting, at each step HDA removes the node with the highest single-layer degree (the maximum of its degree over the two layers), then runs the cascade to convergence and recomputes degrees on the surviving subgraph before the next step. Because the two replicas of a surviving node are always alive together, the inter-layer coupling adds the same constant to every candidate's score and does not affect the ranking, so this is exactly "remove the highest single-layer-degree node." The selected node is removed in both layers and the cascade is run to convergence before the next removal.

HDA on the aggregated graph (multiplex protocol). For the multiplex setting the target is the aggregated (edge-union) graph $\mathcal{A}$ defined in the main paper, where an edge is present iff it appears in at least one layer. HDA ranks by degree in $\mathcal{A}$ and removes the highest-degree node, recomputing degrees on $\mathcal{A}$ after each removal. FINDER is likewise applied to $\mathcal{A}$ treated as an ordinary single-layer graph.

EMD (Effective Multiplex Degree). EMD is an iterative weight-propagation centrality on the multiplex [5]. Let L_j be the set of layers in which node j is present and $M_j = |L_j|$, and let $\deg_\ell(j)$ be the degree of j in layer ℓ. Weights are initialised to the multi-degree $w_j^{(0)} = \sum_{\ell \in L_j} \deg_\ell(j)$ and updated by

$$w_i^{(t+1)} \;=\; \sum_{\ell} \sum_{j \in \mathcal{N}_\ell(i)} \frac{w_j^{(t)}}{\deg_\ell(j)\, M_j}, \tag{I1}$$

re-normalised to sum to one after each sweep, and iterated until the total absolute change falls below 10^{-4} (capped at 30 iterations). The node with the largest weight is removed, after which the weights are recomputed on the surviving subgraph. EMD is the strongest non-learned multilayer heuristic in our comparison.

FINDER (Finding key players). FINDER is a GNN encoder trained with an n-step deep-Q network [6]. We use the authors' *official released implementation* and their published model (trained on Barabási–Albert graphs of $n \in [30, 50]$, matching our own training regime), rather than a re-implementation. To keep the comparison to a pure node-ranking method, FINDER is run with *reinsertion and decycling disabled* — a large part of FINDER's published dismantling strength comes from this post-processing, and enabling it for FINDER while every other method scores raw removal order would be an asymmetric advantage. FINDER removes nodes at StepRatio = 0.01 (the same 1%-batch granularity used for every method here), stops at the 10%-LCC threshold, and its curve is integrated by the same Simpson rule and N_0^2 normalisation as all other methods. Under this configuration FINDER attains mean AUC 0.0919 on the 53-network set (our earlier self-reimplementation, weaker, gave 0.1162); Ours remains lowest at 0.0878. For the multiplex protocol FINDER is applied to the aggregated graph $\mathcal{A}$ treated as an ordinary single-layer graph.

MD (MultiDismantler). MD is the multilayer-specialised DRL dismantler of Gu et al. [3] (*Nat. Mach. Intell.* 2025), the current SOTA on interdependent network dismantling. We use the authors' released model, trained *directly* on synthetic multilayer BA interdependent graphs under the cascade-failure reward — i.e. MD sees the multilayer cascade dynamics at training time, whereas Ours and Disco103 are transferred zero-shot from single-layer training. On the interdependent (GMCC) protocol MD is evaluated in its native setting; on the multiplex (GACC) protocol its trained policy is applied zero-shot. In both protocols MD removes one node per step, runs the cascade (GMCC) or recomputes aggregated connectivity (GACC) to convergence, stops at the 10% threshold, and is scored with the same Simpson/n_{per}^2 convention as every other method, so the only quantity compared is removal order.

Appendix J Integration rule and threshold sensitivity

The main-paper AUC uses Simpson integration of the LCC curve down to the 10%-LCC threshold. To confirm the single-layer ranking is not an artefact of these two choices, we recompute the 53-network means under all combinations of three integration rules (left-rectangle/Riemann, trapezoid, Simpson) and three stopping thresholds (1%, 5%, 10%), reusing the same dismantling trajectories (batch 1%, N_0^2 normalisation). Table J1 reports Ours, Disco103 and HDA in each of the nine variants.

Table J1 Integration-rule × threshold sensitivity on the 53 single-layer networks (mean AUC, lower is better). Ours ranks best in all 9/9 variants; the Ours−Disco103 and Ours−HDA gaps are essentially constant (≈ -0.0036 and -0.0028). These absolute values use the m=1 agent (0.0884 at Simpson/10%); the main-paper headline uses the M= 2 agent (0.0878), which only widens the gaps. Changing the integration rule barely moves any number; changing the threshold shifts all methods together and leaves the ordering unchanged.

Integration	Threshold	Ours	Disco103	HDA
Left-rect.	1%	0.0907	0.0943	0.0935
Left-rect.	5%	0.0897	0.0934	0.0925
Left-rect.	10%	0.0889	0.0925	0.0917
Trapezoid	1%	0.0901	0.0937	0.0929
Trapezoid	5%	0.0891	0.0928	0.0919
Trapezoid	10%	**0.0884**	0.0919	0.0911
Simpson	1%	0.0901	0.0937	0.0929
Simpson	5%	0.0891	0.0927	0.0919
Simpson	10%	**0.0884**	0.0919	0.0911

Appendix K PPO hyper-parameter grid

Section 6.6 of the main paper reports PPO as a near-failure inner optimiser (single-layer eval AUC 0.1321). To confirm this is not an artefact of an unlucky hyper-parameter choice, we sweep a 3×3 grid over learning rate $\{1, 3, 10\} \times 10^{-4}$ and clip $\{0.1, 0.2, 0.3\}$ (value coefficient 0.05, K=4 epochs fixed), training 3 seeds per cell for 500 inner steps and selecting the best seed per cell by the same 3-network screen (Crime/HI-II-14/Enron) used for the other RL baselines. Table K1 reports the resulting full-evaluation AUC. The best configuration (lr = 10^{-4}, clip 0.3) reaches 0.2359—still 2.7× worse than Ours (0.0884 at this m=1 setting) and worse than both REINFORCE (0.0894) and HDA (0.0911). PPO is thus a genuine near-failure inner optimiser under the meta-net's short-episode regime, not a mistuned straw man.

Table K1 PPO inner-optimiser grid: best-seed single-layer eval AUC (lower is better) for each learning-rate × clip combination. Every cell is far above Ours (0.0884); the best cell is bold. Note that the cell reproducing the main-paper PPO setting (lr 3×10^{-4}, clip 0.2) reads 0.3602 here rather than the reported 0.1321: the main text selects best-of-10 seeds whereas this grid uses best-of-3, so the grid values are systematically inflated by the smaller seed budget and are reported only to show that *no* configuration rescues PPO.

lr \ clip	0.1	0.2	0.3
1×10^{-4}	0.2704	0.2988	**0.2359**
3×10^{-4}	0.3726	0.3602	0.3382
1×10^{-3}	0.2650	0.3110	0.4015

Appendix L Full head-to-head significance table

Table L1 gives the complete paired head-to-head statistics behind the significance statements in the main paper. Each row pairs Ours against one baseline across the networks common to both methods: diff = $\text{AUC}_{\text{base}} - \text{AUC}_{\text{ours}}$ (positive ⇒ Ours better, since lower AUC is better). The bootstrap column is a percentile 95% CI on the mean diff (10000 resamples, seed 12345); ties use $|\text{diff}| < 10^{-9}$. The single-layer pick is the M= 2 agent (0.0878); the multilayer picks are the differentiable-MC M= 2 agent (GMCC 0.1310, GACC 0.2200). `disco103_fair` is the BAML-matched Disco103 and `disco103_orig` is the original released model.

Table L1 Complete paired head-to-head significance (Ours vs. each baseline). W/L/T = win/loss/tie; sign and Wilcoxon are two-sided paired p-values. Ours is significantly better wherever the CI excludes zero; the three statistical ties (GACC vs. Disco103, GMCC vs. EMD, GMCC vs. MD) are reported honestly.

Benchmark	Comparison	W/L/T	$\overline{\text{ours}}$	$\overline{\text{base}}$	$\overline{\text{diff}}$	Boot. 95% CI	sign p	Wilcoxon p
53 single	vs Disco103	42/11/0	0.0878	0.0919	+0.0041	[6.7×10^{-4}, 7.9×10^{-3}]	2.3×10^{-5}	3.7×10^{-4}
53 single	vs HDA	35/18/0	0.0878	0.0911	+0.0033	[1.0×10^{-3}, 5.6×10^{-3}]	0.027	4.4×10^{-3}
GMCC	vs Disco103_fair	25/5/3	0.1310	0.1775	+0.0464	[0.027, 0.068]	3.3×10^{-4}	2.4×10^{-6}
GMCC	vs Disco103_orig	27/6/0	0.1310	0.1589	+0.0279	[0.015, 0.043]	3.2×10^{-4}	4.2×10^{-5}
GMCC	vs HDA	25/7/1	0.1310	0.1523	+0.0212	[0.011, 0.035]	2.1×10^{-3}	5.4×10^{-5}
GMCC	vs EMD	16/11/6	0.1310	0.1324	+0.0014	[-3.2×10^{-3}, 6.5×10^{-3}]	0.44	0.56
GMCC	vs MD	15/15/3	0.1310	0.1296	−0.0014	[-6.4×10^{-3}, 3.4×10^{-3}]	1.00	0.90
GACC	vs MD	33/0/0	0.2200	0.3203	+0.1003	[0.080, 0.123]	2.3×10^{-10}	2.3×10^{-10}
GACC	vs EMD	33/0/0	0.2200	0.2598	+0.0398	[0.031, 0.049]	2.3×10^{-10}	2.3×10^{-10}
GACC	vs FINDER	24/8/1	0.2200	0.2323	+0.0123	[6.9×10^{-3}, 0.018]	7.0×10^{-3}	5.4×10^{-5}
GACC	vs HDA	23/8/2	0.2200	0.2323	+0.0123	[7.0×10^{-3}, 0.018]	0.011	4.0×10^{-5}
GACC	vs Disco103	14/16/3	0.2200	0.2204	$+4.3\times10^{-4}$	[-2.9×10^{-3}, 4.2×10^{-3}]	0.86	0.70

Table M1 Hand-designed RL rules versus meta-net-trained task agents on 53 single-layer networks. All rows share the identical DiscoGNNAgent encoder, training distribution, 10 random seeds, and BA-validation selection protocol; only the update rule differs. The lowest mean ANC is shown in bold and the second-lowest is underlined.

Update rule	Trainable signal	Mean ANC ↓	vs. HDA	vs. Ours (155K)
REINFORCE	policy gradient + mean baseline	0.0894	36/17	21/32
PPO	clipped surrogate + value baseline	0.1321	6/47	7/46
HDA	static degree heuristic	0.0911	—	18/35
Disco103 meta	DeepMind 754K meta-net	0.0919	23/30	11/42
Ours meta-net (155K)	graph-domain meta-net	<u>0.0878</u>	**35/18**	—

Ours is significantly better than every baseline on the 53 single-layer networks and on both multilayer protocols against the video-game-domain Disco103, HDA, and the multilayer-specialised MD/EMD/FINDER, with three exceptions that are honest statistical ties: GACC vs. Disco103 ($p = 0.86$, mean gap 4.3×10^{-4} in Ours' favour), GMCC vs. EMD, and GMCC vs. MD. Ours is not significantly worse than any baseline on any protocol.

Appendix M Additional Ablation and Sensitivity Analyses

This appendix provides additional experiments that complement the main ablation study in Section 6.6. We examine alternative hand-designed RL update rules, meta-network capacity, and the recurrent meta-state component.

M.1 Hand-Designed RL Update Rules

The main ablation in Table 6 already shows that replacing the learned meta-update rule with REINFORCE does not reproduce the full performance of the proposed method. We further compare the learned update rule with two classical policy-gradient algorithms, REINFORCE and PPO, under a matched single-layer evaluation protocol. This experiment examines whether the observed gain can be obtained simply by replacing the meta-net with a standard RL optimizer while retaining the same task-agent architecture.

All configurations use the identical DiscoGNNAgent encoder, with 155K trainable parameters, the same BA training distribution ($n \in [30, 50]$, $m = 4$), the same 10 random seeds $\{1000, 1077, \ldots, 1693\}$, the same 500 inner training steps, and the same disjoint BA validation set for task-agent selection. Evaluation is performed on the same 53 single-layer networks using the 10%-GCC-threshold ANC metric. The only difference is the update rule.

REINFORCE achieves a mean ANC of 0.0894, which is competitive with HDA (0.0911) but remains worse than the proposed meta-net (0.0878). PPO performs substantially worse, with a mean ANC of 0.1321: under the short-budget protocol REINFORCE learns reliably on 4 of the 10 seeds, PPO on 1, and the proposed meta-net on at least 9. A full 3×3 grid over PPO learning rate and clip confirms this is not an artefact of poor tuning (Appendix K). These results show that the learned update rule improves not only mean performance but also training robustness.

M.2 Meta-Network Capacity and Architecture

We additionally evaluate the sensitivity of the meta-network to its capacity and recurrent architecture. For the capacity analysis, we consider $p \in \{64, 128, 256\}$. The $p = 64$ variant contains approximately 50K parameters and is competitive on the single-layer benchmark, but degrades by +13.1% on the 33-network multilayer transfer, indicating limited capacity for cross-protocol generalization. The $p = 256$ variant contains approximately 197K parameters and becomes less stable under the current short inner-loop budget, with degradations of approximately +21% on the single-layer benchmark, +8% on GMCC, and +0.9% on GACC. The selected $p = 128$ therefore provides a favorable accuracy–stability trade-off. Full numerical results are reported in Table M2 below.

We also remove the recurrent meta-RNN gate that carries the meta-state across inner steps. Without this component, single-layer ANC degrades from approximately 0.107 to 0.1165, and the variant does not achieve the best result on any of the three evaluated connectivity measures (GCC, GMCC, and GACC). This result supports the role of the recurrent meta-state in maintaining useful information across the inner optimization trajectory. Full results are provided in Table M2 below.

Table M2 Meta-net architecture and capacity ablations under the single-pick protocol (mean AUC, lower is better; "protocols won" counts how many of the three benchmarks the variant leads). Bypassing the meta-RNN gate degrades every benchmark and wins none, showing the recurrent gate is a functional component. The p=128 main configuration (see Table M1 above) outperforms both p=64 and p=256 on two of three protocols; p=256 collapses on single-layer (0.1069 vs. p=128's 0.0884, a +21% degradation) and shows weaker multilayer transfer, suggesting training instability at higher capacity under the present meta-training budget.

Meta-net variant	s53	GMCC	GACC	Protocols won
p=256 (#pick-v1)	0.1069	0.1419	0.2242	0/3
p=256, new seeds (#11)	0.1066	0.1424	0.2267	0/3
p=64 (#4)	0.1129	0.1428	0.2255	0/3
meta-RNN bypass (#7)	0.1165	0.1526	0.2289	0/3
For reference, p=128 (Table M1): 0.0884 / 0.1378 / 0.2223.				

Three conclusions follow. (i) *The meta-RNN gate is necessary.* Bypassing the recurrent gate degrades s53 from ~ 0.107 to 0.1165 (+0.011) and loses on all three protocols, so the discovered rule's lifetime-recurrent state is a working component of the architecture rather than an inert add-on. (ii) p=128 *is the capacity sweet spot.* Both p=64 and p=256 underperform p=128 (0.0884/0.1378/0.2223). p=64 shows mild underfitting on multilayer transfer (0.1428/0.2255 vs. 0.1378/0.2223), while p=256 suffers a severe collapse on single-layer (0.1069 vs. 0.0884, +21%) and weaker multilayer performance. The all-around p=256 failure is atypical for mere overparameterization and suggests training instability at higher capacity under the current meta-training budget; a longer meta-training run or adjusted

hyperparameters may stabilize it. (iii) *Capacity improvements are reproducible.* A fresh p=256 run with new seeds reproduces the reference within < 0.006 on every benchmark, ruling out a lucky-seed explanation.

Appendix N Cross-task seed-to-seed stability

Beyond mean performance (Section 6.6 of the main paper), the two meta-nets differ markedly in how consistently their fresh agents converge across random seeds on the MIS / MVC / MC cross-task setting. Table N1 reports the standard deviation of the evaluation AUC across the five fresh-agent seeds. Ours is 19×/17×/1.6× more stable than Disco103 on MIS/MVC/MC; on MVC in particular, two of Disco103's five seeds collapse to random-level AUC ($\geqslant 0.95$) while all five of ours converge. This lower training variance is consistent with the graph-domain alignment argument of the main paper: a domain-matched update rule yields a smoother fresh-agent optimisation landscape.

Table N1 Seed-to-seed standard deviation of the cross-task evaluation AUC across 5 random fresh-agent seeds (lower is more stable).

Task	Ours std	Disco103 std	Ours advantage
MIS	0.0022	0.0420	**19.2× lower**
MVC	0.0089	0.1531	**17.2× lower**
MC	0.0015	0.0024	1.6× lower

Appendix O Zero-shot scale-up to larger multilayer networks

To evaluate whether the zero-shot transfer reported in Section 6.3 of the main paper holds on substantially larger graphs, we apply the same frozen agents (Ours, trained only on single-layer BA graphs of $n \in [30, 50]$) to three sacchpomb genetic multilayer networks with 668, 1358, and 2324 nodes per layer under the same cascade-failure protocol. Table O1 reports the per-network 10%-GMCC-threshold AUC for MD, EMD, Ours, and HDA.

Table O1 Zero-shot scale-up: three sacchpomb genetic multilayer networks with $n_{\text{per}} = 668/1358/2324$, under cascade failure. Ours stays within 1.09× of the multilayer-specialized MD and separates clearly from HDA on every network; EMD edges slightly ahead of Ours at this larger scale. Per network the lowest ANC is in bold and the second-lowest underlined.

Network (layers)	n_{per}	$\lvert E_{\text{merged}}\rvert$	MD	EMD	**Ours** (zero-shot)	HDA
sacchpomb_genetic-1-4	668	3546	**0.0268**	<u>0.0321</u>	0.0339	0.0471
sacchpomb_genetic-3-4	1358	10309	**0.0152**	<u>0.0170</u>	0.0173	0.0264
sacchpomb_genetic-4-6	2324	46278	<u>0.0910</u>	**0.0898**	0.0937	0.1004
mean	—	—	**0.0443**	<u>0.0463</u>	0.0483	0.0580
ratio vs. MD	—	—	1.00×	1.04×	1.09×	1.31×

Ours, trained only on synthetic single-layer BA graphs of size 30–50, tracks the cascade-specialized MD closely, staying within 1.09× in mean ANC across the three networks ($n_{\text{per}} = 668/1358/2324$), with its relative gap to MD *shrinking* as graphs grow—down to 3% on the largest network and 9% in the mean—while beating HDA on every network. In absolute terms the gap is tiny ($\leqslant 0.007$ per network and 0.004 in mean ANC) and is dominated by graphs whose cascade dynamics produce GCC trajectories a single-layer agent has never seen. That Ours holds within a few percent of the cascade-specialized SOTA, and ahead of HDA, on graphs 50–200× larger than its training size shows that the meta-net's graph-domain priors transfer to larger interdependent networks without re-training.

Appendix P Cross-topology generalization

We evaluate zero-shot transfer of agent_06 (trained solely on BA graphs, $n \in [30, 50]$) to four topology families, holding size fixed to $n \in [30, 50]$ to isolate topological variation. We generate 50 graphs per topology (Barabási–Albert, Erdős–Rényi, Watts–Strogatz, power-law) and report mean AUC on single-layer LCC dismantling (lower is better). *Win/Loss*: number of graphs on which Ours beats / is beaten by HDA.

Table P1 Zero-shot cross-topology generalization. Mean AUC (lower is better); training topology: BA.

Topology	$\lvert G\rvert$	Ours AUC↓	HDA AUC↓	CoreHD AUC↓	CI AUC↓	Win/Loss
BA (train)	50	**0.0069**	0.0070	0.0084	0.0074	35/15
ER	50	**0.0097**	0.0098	0.0121	0.0101	32/18
WS	50	**0.0100**	0.0101	0.0114	0.0093	29/21
Power-law	50	**0.0073**	0.0074	0.0089	0.0078	34/16

Ours maintains a mean-AUC advantage over HDA on all four topologies, with win rates of 58%–70%. On Watts–Strogatz graphs (near-regular small-world), CI is competitive (AUC 0.0093 < 0.0100) because CI was designed for that regime; Ours still beats HDA on 29/50 graphs. These results indicate that the meta-net's scoring criterion generalises to structural patterns absent from BA training data.

Table Q1 Kendall-τ between agent_06 node scores and classical centralities (100 BA graphs, $n \in [30, 300]$). Higher $|\tau|$ means stronger rank agreement.

	Degree	Betweenness	CI	Closeness	Eigenvector	k-core
τ	**0.924**	0.835	0.737	0.561	0.530	0.206

Appendix Q Centrality alignment

To understand what scoring criterion the agent has internalized, we measure the Kendall rank correlation (τ) between the agent's node scores and six classical centrality measures on 100 BA graphs ($n \in [30, 300]$, 50 each at $n = 30$ and $n = 300$).

The agent's ranking correlates most strongly with degree ($\tau = 0.924$) and betweenness ($\tau = 0.835$), suggesting it has learned to integrate local connectivity and global reachability. The low correlation with k-core ($\tau = 0.206$) shows the agent does not simply copy any single classical heuristic.

Appendix R Gradient-based saliency analysis

We compute the gradient of the top-scored node's score with respect to each entry of the input adjacency matrix, $|\partial \text{score}/\partial A_{ij}|$, and bucket entries into three groups: diagonal (self-loops), existing edges (direct), and absent potential edges (two-hop structural holes). We average over 60 BA graphs ($n = 30$–50).

Table R1 Mean gradient magnitude bucketed by adjacency position (60 BA graphs). *direct/twohop* < 1 implies the agent is more sensitive to absent than to present edges.

Graphs	$\lvert\nabla\rvert_{\text{self}}$	$\lvert\nabla\rvert_{\text{direct}}$	$\lvert\nabla\rvert_{\text{twohop}}$	direct/twohop
All (60)	1.54×10^{-3}	1.13×10^{-3}	1.73×10^{-3}	**0.66**

The agent is more sensitive to absent two-hop connections than to present edges (ratio $0.66 < 1$), indicating it implicitly evaluates structural holes — that is, it considers what would be disconnected if a candidate node were removed, rather than only its immediate neighbourhood. This behaviour is consistent with the high betweenness correlation ($\tau = 0.835$) in Table Q1.

Table R2 decomposes the same gradient by hop distance from the top-scored node. Because the GNN uses three message-passing layers, its receptive field spans three hops, so gradients vanish beyond 3-hop by construction; the informative quantity is how mass is distributed *within* that field. The 2-hop band (excluding the 1-hop ring) carries the largest share in every network (41–61%, mean 52%), while the 1-hop band is the smallest (mean 12%), confirming that decisions are governed by local structural context rather than by the node's immediate contacts, and are therefore invariant to graph size.

Table R2 Gradient magnitude by hop distance from the top-scored node, as a fraction of the total 0–3 hop gradient, on three canonical networks. n is the number of nodes; the 2/3-hop bands together account for a mean 83% of the signal.

Network	n	0-hop	1-hop	2-hop	3-hop
karate	34	0.03	0.06	**0.61**	0.30
florentine	15	0.08	0.17	**0.41**	0.34
davis	32	0.03	0.13	**0.53**	0.31
mean	—	0.05	0.12	**0.52**	0.32

Appendix S Scalability benchmark

We measure wall-clock time for single-layer LCC dismantling on BA graphs ranging from $n = 100$ to $n = 10000$ using agent_06 in greedy one-at-a-time mode (CPU, no GPU). Each size is repeated three times; means are reported.

Table S1 Wall-clock scalability. *Single fwd*: one GNN forward pass (ms). *Total*: full greedy dismantling to 10% LCC. AUC lower is better (near-zero at large n indicates all methods reach near-optimal on power-law BA graphs). Ours/HDA: ratio of total times.

n	Fwd (ms)	Ours AUC↓	HDA AUC↓	CHD AUC↓	Ours (s)	HDA (s)	Ours/HDA
100	3.9	0.0025	0.0026	0.0032	0.03	0.000	95×
500	2.4	0.0005	0.0005	0.0008	0.38	0.007	58×
1000	3.1	0.0002	0.0002	0.0004	1.38	0.024	58×
2000	9.8	0.0001	0.0001	0.0002	3.28	0.075	44×
5000	23.6	<0.0001	<0.0001	<0.0001	74.96	0.702	107×
10000	155	<0.0001	<0.0001	<0.0001	490.80	3.096	159×

A single forward pass takes only 155 ms at $n = 10000$. The total dismantling time grows super-linearly because the GNN is re-evaluated after each node removal; a batch-removal variant (removing $n/50$ nodes per step, ≈ 50 steps) would reduce the $n = 10000$ runtime to roughly $50 \times 0.155 \approx 8$ s. AUC approaches zero for $n \geqslant 5000$ on BA graphs because the power-law degree distribution makes any degree-correlated method near-optimal; the advantage of Ours over HDA and CoreHD is most pronounced on the real-world networks in Tables 1–3 of the main paper, not on large synthetic BA graphs.

Appendix T Topological correlates of MD's advantage (GMCC, 33 networks)

To understand *when* the specialized MultiDismantler (MD) outperforms Ours on the GMCC protocol, we compute the Spearman rank correlation between six topological indices and $\Delta = \mathrm{AUC}_{\mathrm{Ours}} - \mathrm{AUC}_{\mathrm{MD}}$ (positive $\Rightarrow$ MD wins, i.e. Ours is relatively weaker) across the 33 multilayer interdependent networks.

Table T1 Spearman ρ between topological index and MD's advantage ($\Delta = \mathrm{AUC}_{\mathrm{Ours}} - \mathrm{AUC}_{\mathrm{MD}}$, positive = MD better) on the 33 GMCC networks ($N = 33$). Only the two marginally significant factors are highlighted; four of six indices show no significant correlation ($p > 0.10$), indicating cross-topology stability.

Topological index	Spearman ρ	p-value	N
Degree-distribution entropy	+0.179	0.3182	33
Clustering coefficient	**−0.334**	**0.0575**	33
Assortativity	−0.038	0.8317	33
Average degree	−0.280	0.1141	33
Density	−0.253	0.1551	33
Gini coefficient	**+0.341**	**0.0523**	33

Four of six indices are non-significant ($p > 0.10$), showing that the Ours–MD performance gap is largely topology-independent; MD wins on 15/33 networks (45.5%), close to parity. Two marginal trends provide preliminary "when" guidance: high clustering ($\rho = -0.334$, $p = 0.0575$) favours Ours, while high degree heterogeneity / hub-domination (Gini $\rho = +0.341$, $p = 0.0523$) favours MD. These findings are discussed in Section 7.2 of the main paper.

Appendix U Structural correlates of Ours' advantage (single-layer, 53 networks)

On the 53 single-layer real networks we compute the Spearman rank correlation between six topological indices and $\Delta = \mathrm{AUC}_{\mathrm{HDA}} - \mathrm{AUC}_{\mathrm{Ours}}$ (positive $\Rightarrow$ Ours better). The most informative finding is for average degree:

Table U1 Spearman ρ between average degree and Ours' advantage over HDA on the 53 single-layer real networks ($N = 53$).

Topological index	Spearman ρ	p-value	N
Average degree	**+0.282**	**0.0410**	53

Ours shows a significantly larger advantage on high-average-degree networks ($\rho = +0.282$, $p = 0.041$), consistent with the average-degree correlation analysis in Appendix U above. In contrast, the multilayer GMCC correlation between average degree and MD's advantage is weaker ($\rho = -0.280$, $p = 0.1141$; Table T1), reflecting the different role of degree in single-layer versus interdependent-cascade protocols.

References

1 Artime O, et al. Robustness and resilience of complex networks. Nat Rev Phys, 2024, 6: 114–131
2 Oh J, Farquhar G, Kemaev I, et al. Discovering state-of-the-art reinforcement learning algorithms. Nature, 2025, 648: 312–319
3 Gu W, Yang C, Li L, Hou J, Radicchi F. Deep-learning-aided dismantling of interdependent networks. Nat Mach Intell, 2025, 7: 1–12
4 Albert R, Jeong H, Barabási A L. Error and attack tolerance of complex networks. Nature, 2000, 406: 378–382
5 Osat S, Faqeeh A, Radicchi F. Optimal percolation on multiplex networks. Nat Commun, 2017, 8: 1540
6 Fan C, Zeng L, Sun Y, Liu Y Y. Finding key players in complex networks through deep reinforcement learning. Nat Mach Intell, 2020, 2: 317–324
7 Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. ArXiv:1707.06347, 2017