

Deep-learning-aided dismantling of interdependent networks

Received: 26 September 2024

Accepted: 4 June 2025

Published online: 14 July 2025

 Check for updates

Weiwei Gu¹, Chen Yang¹✉, Lei Li², Jinqiang Hou¹ & Filippo Radicchi¹³✉

Identifying the minimal set of nodes whose removal breaks a complex network apart, also referred as the network dismantling problem, is a highly non-trivial task with applications in multiple domains. Whereas network dismantling has been extensively studied over the past decade, research has primarily focused on the optimization problem for single-layer networks, neglecting that many, if not all, real networks display multiple layers of interdependent interactions. In such networks, the optimization problem is fundamentally different as the effect of removing nodes propagates within and across layers in a way that can not be predicted using a single-layer perspective. Here we propose a dismantling algorithm named MultiDismantler, which leverages multiplex network representation and deep reinforcement learning to optimally dismantle multilayer interdependent networks. MultiDismantler is trained on small synthetic graphs; when applied to large, either real or synthetic, networks, it displays exceptional dismantling performance, clearly outperforming all existing benchmark algorithms. We show that MultiDismantler is effective in guiding strategies for the containment of diseases in social networks characterized by multiple layers of social interactions. Also, we show that MultiDismantler is useful in the design of protocols aimed at delaying the onset of cascading failures in interdependent critical infrastructures.

Networks are ubiquitous in modelling both natural (for example, climate networks and protein–protein interaction networks) and man-made (for example, social networks and infrastructural networks) complex systems^{1,2}. Connectedness is essential for the overall network function, as being part of the same connected component is a necessary condition for interaction³. Due to their heterogeneous structure, connectedness in real networks hinges on specific sets of nodes. Identifying the minimal set of nodes whose removal would destroy any extensively connected components is known as the network dismantling problem⁴. Network dismantling has numerous applications: optimally inhibiting specific enzymes or proteins for drug design is a network dismantling problem⁵; enhancing the tolerance and robustness of a small set of nodes, such as traffic or electronic sites, to maximally improve the efficiency of infrastructural networks can be thought as

network dismantling^{6,7}; finding the smallest set of people that should be vaccinated to prevent a disease outbreak is well approximated by the solution of a network dismantling problem^{4,8}.

Dismantling is among the most fundamental optimization problems that can be formulated on a network, but it is, unfortunately, NP-hard (non-deterministic polynomial time)^{9,10}. As such, exact solutions can be obtained on small networks only. In medium and large networks, such as those typically considered in applications, solutions can only be approximated. Many existing dismantling algorithms provide approximate solutions that are based on node-centrality heuristics, such as high degree adaptive (HDA)¹¹, collective influence (CI)⁴ and MinSum⁹. These heuristic methods do not optimize a global function, provide no performance guarantees and lack generalization capabilities. Machine learning-based attack strategies such as the graph

¹College of Information Science and Technology, Beijing University of Chemical Technology, Beijing, People's Republic of China. ²School of Software Engineering, University of Science and Technology of China, Hefei, People's Republic of China. ³Center for Complex Networks and Systems Research, Luddy School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN, USA. ✉e-mail: chenyang@buct.edu.cn; filiradi@iu.edu

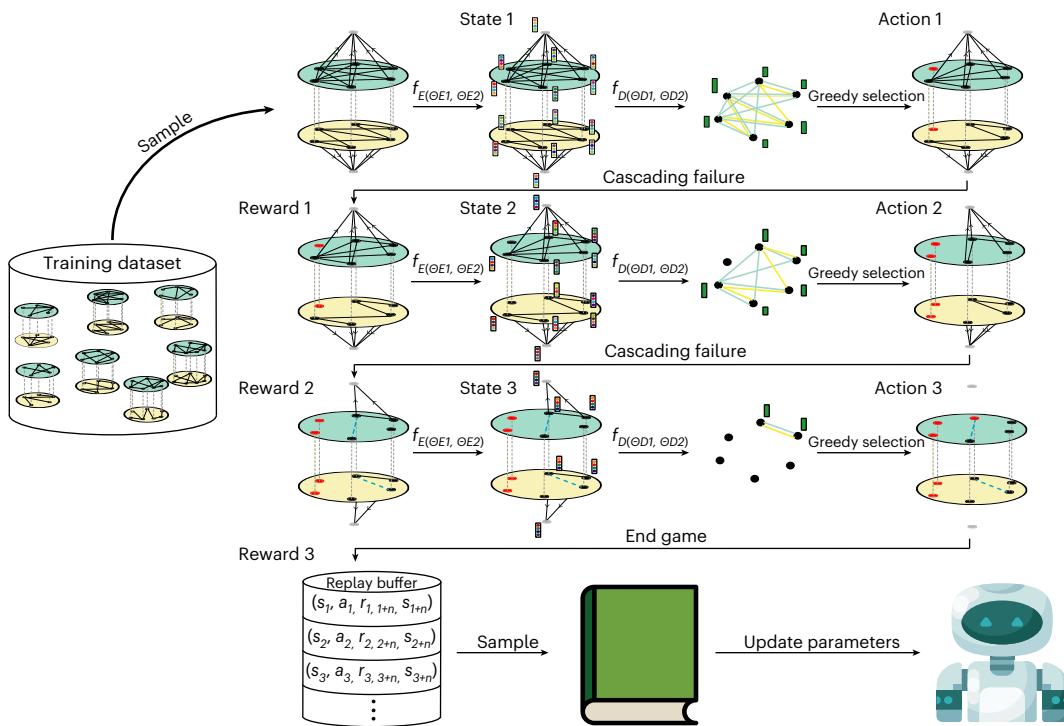


Fig. 1 | The training process of MultiDismantler. The process includes generating training networks, dismantling such networks and learning from previous experience. The training data includes over 20,000 synthetic interdependent networks, with sizes ranging from 30 to 50 nodes, generated according to the geometric multiplex model. To disintegrate a network, we first add a virtual node (denoted in grey) to each layer; all nodes in the layer are neighbours of the virtual nodes, but not vice versa. With the MGNN model, which consists of a graph convolutional neural network model (θ_{E1}) and node-level interlayer attention mechanism (θ_{E2}), MultiDismantler encodes multiplex networks, including the embedding of virtual nodes as well as the final embedding of each node. Subsequently, the agent decodes the embedding vectors using MLPs (θ_{D1}) and layer-level attention mechanism (θ_{D2}) to compute the expected return (Q value) of each node, which reflects the expected

reward that can be obtained by removing a node under the current network conditions. The agent uses an ϵ -greedy strategy to remove nodes (red nodes): with probability $1 - \epsilon$, it selects the node with the highest Q value; otherwise, it randomly chooses the removal node. Considering the interdependencies between layers in a multiplex network, a cascading failure may occur after the agent removes a node, leading to the removal of more edges (blue lines). After the node deletion operation, the agent reencodes and decodes the residual network, ranking the importance of each node, until the network is completely dismantled. Once the network is fully dismantled, we collect the n -step information in the form of $(s_t, a_t, r_{t:t+n}, s_{t+n})$ tuples and store them in an experience replay buffer. The agent learns by randomly sampling mini-batches from the experience replay buffer to update its parameters.

dismantling with machine learning (GDM)¹² and the neural influence ranking model (NIRM)¹³ learn dismantling policies from exact solutions on small synthetic such as Barabási–Albert (BA) and Erdős–Rényi (ER) networks. FINDER¹⁴ eliminates the need for direct supervision by using a reinforcement learning framework, where the agent learns optimal node removal strategies through trial-and-error interactions with the network environment. These policies appear successful when used to approximate solutions of large-scale dismantling problems.

The aforementioned algorithms focus solely on single, isolated networks, neglecting the fact that real networks are often formed by multiple interdependent layers¹⁵. A network composed of interdependent layers is more vulnerable than each of its layers considered in isolation. Localized damage of the nodes in one layer may lead to the failure of the nodes in the other layers, causing cascading failures throughout the entire system¹⁶. This cascading mechanism makes system collapse difficult to anticipate and control¹⁷. Examples include the 2003 blackout in Italy¹⁶ and the sudden disruptions of the global supply chain during the COVID-19 pandemic in 2020¹⁸. Identifying the minimal set of nodes whose removal leads a multilayer network to collapse is crucial for all these real systems; however, the dismantling problem of a network composed of interdependent layers cannot simply be solved by considering each of its layers in isolation^{19–21}. Currently, two methods, effective multiplex degree (EMD)²⁰ and CoreHLDA²², are specifically designed for the dismantling of multilayer networks. However, they are both based on node-centrality heuristics and are thus affected by

the limitations typical of this class of methods. As we will show here, they often lead to suboptimal dismantling performances compared with our proposed method.

Motivated by recent advancements in applying machine learning techniques to graph optimization problems^{23–25} and building on the foundational framework of FINDER¹⁴, we introduce here a deep-learning-aided dismantling framework named MultiDismantler, which can approximate solutions in large-scale networks with high effectiveness. MultiDismantler first generates a large number of small synthetic networks using the geometric multiplex model²⁶, overcoming the limitations of the BA model, which cannot characterize interactions and correlations across different network layers. Geometric correlations among network layers are ubiquitously present in real multilayer networks²⁷; those correlations are known to dramatically affect the outcome of both structural and dynamical processes occurring on the networks^{28–30}. After training corpus generation, MultiDismantler designs a graph convolutional neural network model with a flexible node-level interlayer attention mechanism to encode nodes and states in multilayer networks. This is followed by an n -step Deep Q-Network (DQN)³¹, which takes node states as input and aims to automatically learn an optimization strategy that quickly dismantles multilayer networks.

MultiDismantler, trained on small synthetic graphs, displays excellent performance across various complex real-world networks, handling arbitrary removal node costs³². We validate MultiDismantler in two practical applications of network dismantling. First,

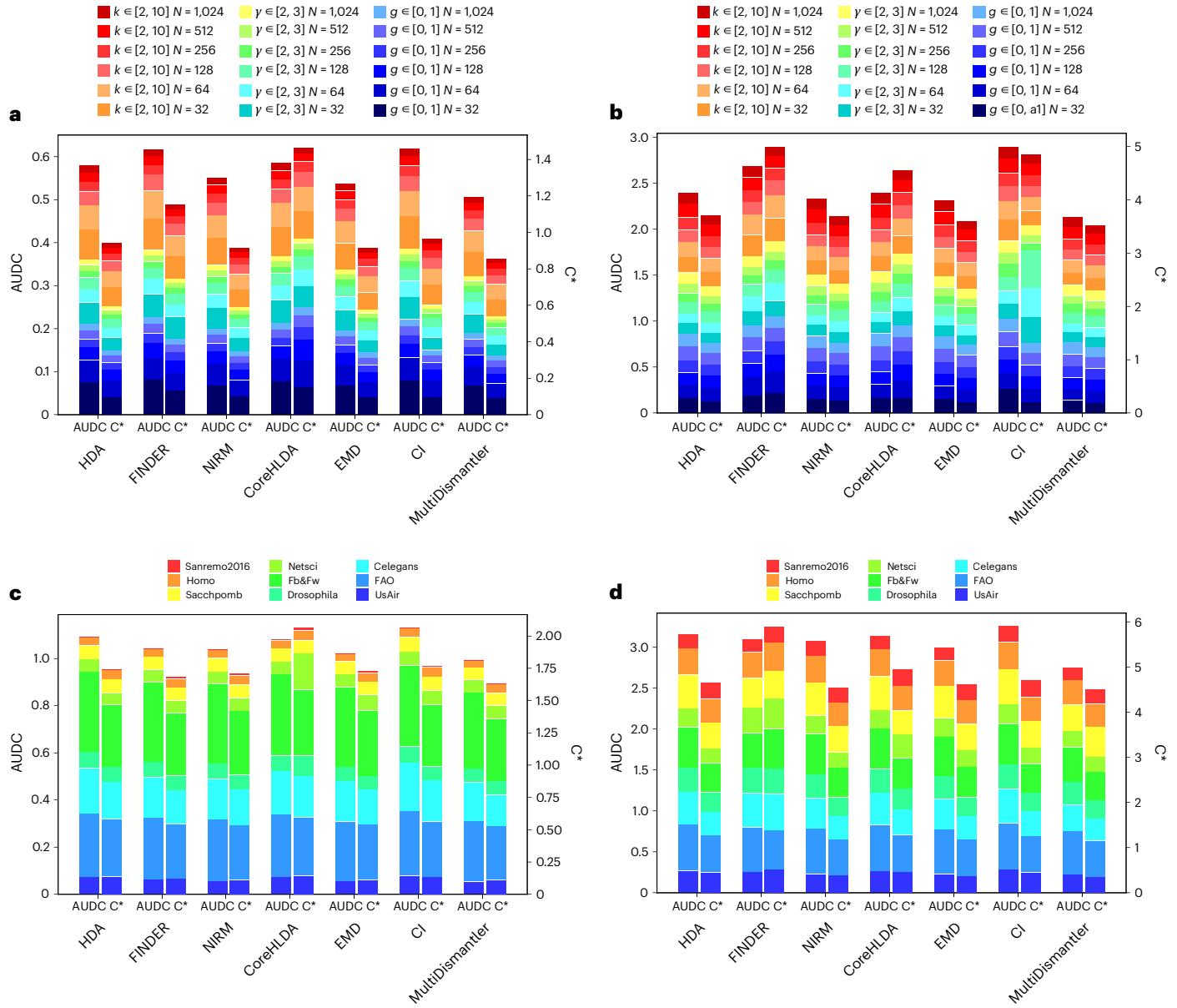


Fig. 2 | Dismantling performance of synthetic and real-world interdependent networks under the metrics AUDC and C' . For synthetic networks, we consider three controllable parameters, that is, interlayer similarity correlation (g), power-law degree distribution exponent (y) and expected mean degree (\bar{k}). We tune one parameter while keeping the other two parameters invariant to generate 20 interconnected networks with various properties (see Supplementary Table 3 for the parameter setting of the GMM model). For example, to generate networks under condition $g \in [0, 1], N = 1,024$, we set $y \in 2.5$ and $\bar{k} \in 6$, and randomly

generate g values from 0 to 1. **a**, The dismantling performance of unit removal cost under AUDC, which quantifies the cumulative AUDC, and C' , which measures the total cost to disintegrate the network smaller than the square root of the original LMCC. **b**, The dismantling performance of degree removal cost. **c,d**, Similarly, for real-world interdependent networks from various domains, we compare the dismantling performance of different algorithms under both unit removal (**c**) and degree removal cost (**d**). See Supplementary Tables 10–12 for the detailed numerical results.

we use MultiDismantler to design optimal immunization strategies in spreading processes on multilayer social networks; second, we take advantage of MultiDismantler as a tool to mitigate system collapse in interdependent networks. In both applications, MultiDismantler demonstrates superior performance compared with all other algorithms existing in the market.

Results

Dismantling interdependent networks

We consider a network consisting of two layers of interactions. Nodes in the two layers are one-to-one interdependent. For simplicity of description but without loss of generality, we do not make a formal distinction between one node and its replica in the other layer; instead, we act as if

the two layers share the same set of nodes. Depending on the context, networks of this type, that is, with distinguishable layers of interaction but indistinguishable nodes across the layers, are also referred to as multiplex networks or edge-coloured graphs^{33,34}. Two nodes are in the same mutually connected component if they are connected by paths in all layers, and all nodes along these paths are also in the same component¹⁶. The largest mutually connected component (LMCC) is the largest of such mutually connected components.

The network dismantling problem involves identifying the optimal sequence of nodes whose removal efficiently results in the fragmentation of the network into mutually connected components of non-extensive size^{19,35} (see ‘Problem formulation’ section in the Methods for the definition of the optimization problem). The removal of

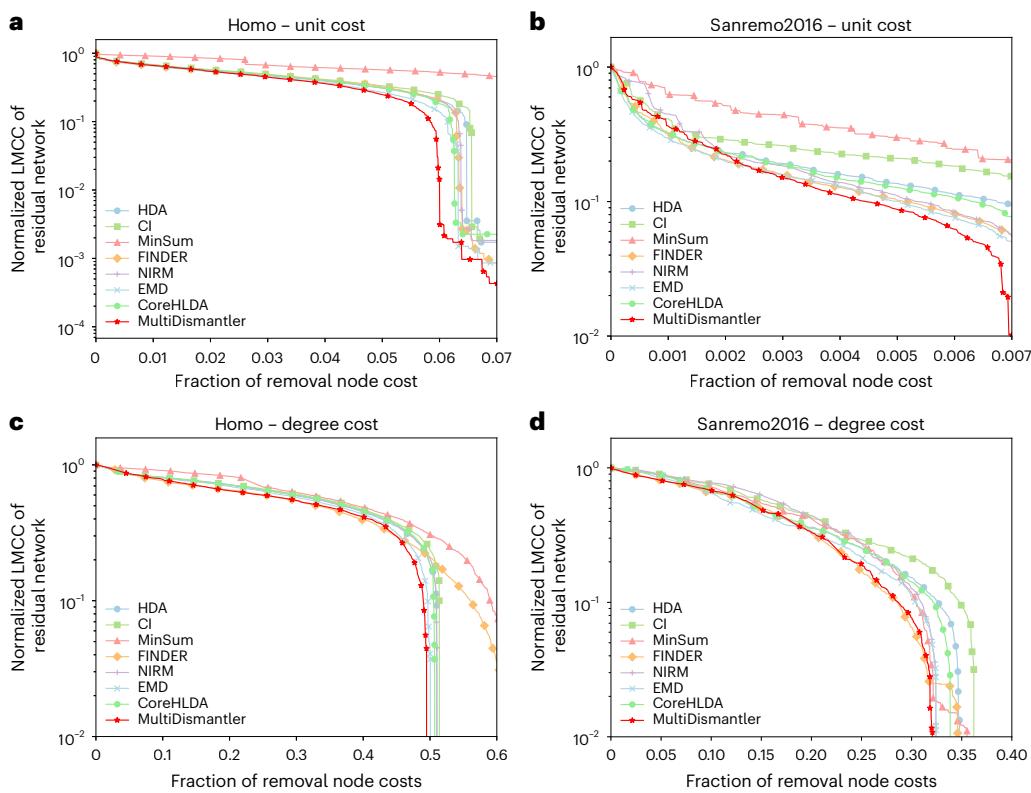


Fig. 3 | The dismantling process of two real-world multiplex networks by different dismantling algorithms. **a–d**, The horizontal axis represents the fraction of removal node costs and the vertical axis represents the normalized

LMCC of the residual network. The dismantling of the Homo and Sanremo2016 networks under unit cost (**a** and **b**) and the dismantling results under degree cost (**c** and **d**) are shown.

each node in the sequence incurs a cost³². We consider two scenarios: the cost of each node equals one, that is, unit cost, and the cost of a node equals the number of edges that are removed together with the node, that is, degree cost. Also, we relax scenario 1 by allowing nodes to have costs taken at random from the uniform, normal and Poisson distributions. This scenario is used to test the generalizability of our algorithm to cases where removal costs are uncertain. The size of the LMCC provides a natural metric to monitor the fragmentation of the network into non-extensive mutually connected components and typically decreases with the cumulative cost of the removed nodes. The area under the dismantling curve (AUDC), that is, the relative size of the LMCC as a function of the relative cost of the removed nodes, is a natural quantity to assess the quality of a given sequence^{12,14}. Another natural metric to assess the quality of a specific sequence of nodes provided by a dismantling algorithm is given by the relative cost required to make the LMCC non-extensive. The dismantling cost, namely C , is given by the smallest value of the cost, normalized by the cost of removing all nodes, required to make the LMCC smaller than the square root of the original LMCC, that is, the conventional threshold used to determine whether a component is non-extensive³⁶.

Proposed approach to the dismantling of interdependent networks

MultiDismantler is the first approach able to integrate multiplex network representation learning with a deep reinforcement learning (DRL) architecture to solve optimization problems on multilayer interdependent networks. To ensure success, three main challenges are addressed by MultiDismantler:

- (1) MultiDismantler is trained on synthetic graphs constructed via a generative network model able to accurately reproduce the structure of real-world interdependent networks. This approach

is essential for mimicking the diversity of real-world networks with synthetic networks, thereby increasing the generalizability of MultiDismantler.

- (2) MultiDismantler relies on an encoding framework designed to effectively capture and preserve both intra- and interlayer network structures and properties. This ensures high-quality representations that facilitate learning and decision-making.
- (3) MultiDismantler appropriately bridges multilayer representation learning vectors with DRL components. This involves defining the network state, specifying node deletion actions, designing the environment and establishing evaluation metrics to guide the learning process.

A schematic illustration of the MultiDismantler framework is provided in Fig. 1, and details on its implementation are reported in ‘MultiDismantler’ section in the Methods. Below, we briefly describe its three main components.

GMM generates synthetic multiplex networks with cross-layer geometric correlations. GMM has been widely used to characterize the properties of real multilayer networks^{26,28}. The model has several tunable parameters controlling both intra- and interlayer properties of the generated network. In particular, the interlayer similarity correlation g plays a vital role in network disintegration prediction²⁸.

Traditional representation learning frameworks for multilayer networks, often utilizing graph neural network architectures, use meta-path or layer-level attention mechanisms to encode nodes across layers^{37–39}. These frameworks independently learn node representations in each layer and concatenate them using deep learning techniques such as multilayer perceptions; however, they fail to fully capture the complexity of the interlayer topology. Here, we propose a new multiplex graph neural network (MGNN) model, which uses a series of layer-specific graph convolution layers to obtain intralayer representations by aggregating information from neighbouring nodes during the message-propagation

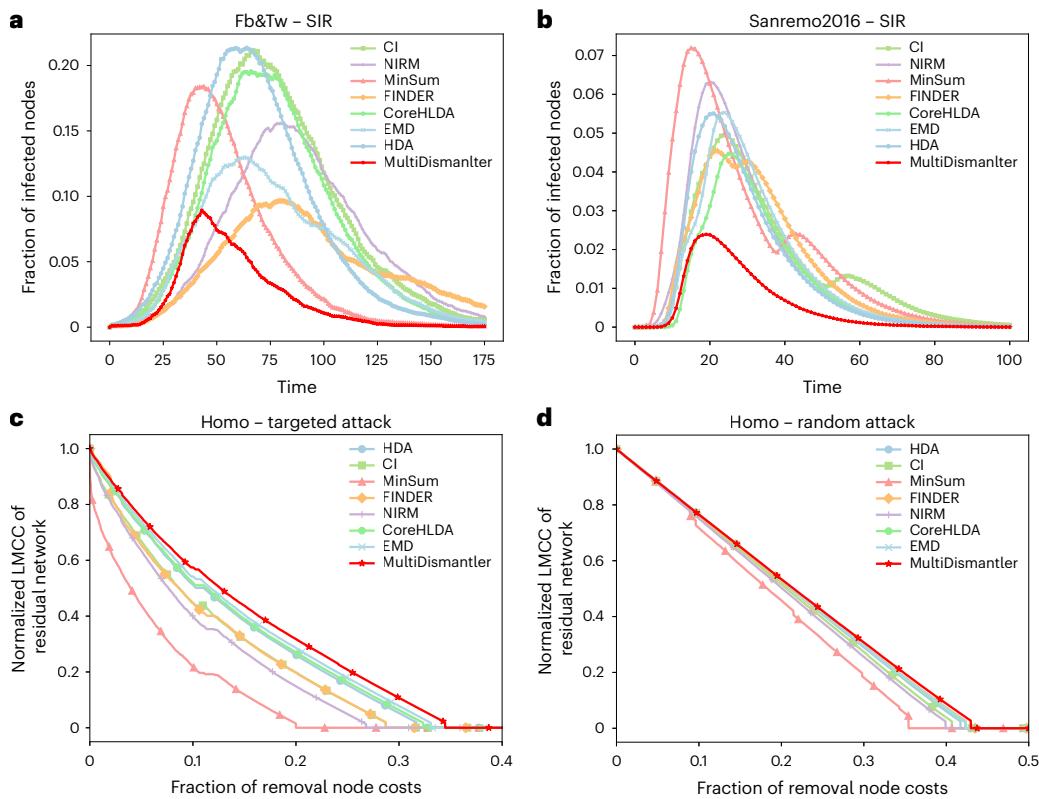


Fig. 4 | Preventing disease spreading and maintaining network robustness with MultiDismantler. **a,b**, MultiDismantler’s performance in reducing the outbreak size of disease on offline and online social networks. We simulate SIR dynamics on the remaining social networks after the node removal strategies of different dismantling algorithms to mimic quarantine or immunization of real-world disease control measures. We display the fraction of nodes in the network that are infected at a specific time during the spreading dynamics. The outbreak size of the entire spreading process is given by the area under the curve. With the

same number of removed nodes, the outbreak size under the MultiDismantler controlling strategies is consistently smaller than that under other comparison algorithms. **c,d**, Simulations of network robustness of the Homo network under targeted attack and random failure. We show the relative size of the LMCC as a function of the relative cost of the removed nodes. Multiplex networks are robust against random failure, and MultiDismantler has a slight advantage in maintaining network robustness.

process. After intralayer propagation, MGNN applies a multifaceted linear transformation matrix to project intralayer representations into a shared representation space. Incorporating feature influence from cross-layer neighbours can enrich node representations; however, the impact of interlayer neighbours on different nodes varies dramatically. Some nodes are influenced primarily by intralayer connections, while others rely more on interlayer edge information. MGNN incorporates a node-level attention mechanism that computes customized attention weights for each node. The final representation of a node in a given layer is a weighted combination of its neighbours’ representations from both intralayer and interlayer connections.

In the context of interdependent network dismantling using DRL, the agent is trained to learn the nodes’ removal strategy that maximally disrupts network connectivity across the layers. The DQN approach addresses this problem by approximating the Q -value function with a deep neural network. The function $Q(s, a; \theta)$ estimates the expected cumulative reward of taking action a in state s with parameters θ . Here, the action a represents the selection of a node to remove, and we use node vectors learned from multiplex representation as action embeddings. The state s represents the current configuration of the multilayer network. We introduce virtual nodes¹⁴, where all nodes are neighbours of the virtual node, and use the vector of virtual node as the state embedding. The reward r is designed to quantify the impact of node removal on the network’s connectivity; we measure rewards using the decrease in the size of the LMCC.

Removing a single node has a long-term influence on future states and actions. To capture this long-term effect, we apply the n -step DQN model³¹. This model aims to minimize the difference between the

predicted Q value and the target Q value, which includes the cumulative immediate reward and the discounted maximum future Q value. The agent removes nodes according to the predicted Q values. As the agent removes nodes and the topology of the multiplex network changes, the embedding vectors of state and action are continuously updated. The node removal process continues until the interdependent network is fully dismantled.

Comparative analysis with FINDER

MultiDismantler builds on foundational ideas from FINDER. Both methods use a deep Q -learning model for network dismantling; also, they rely on common techniques, such as virtual node representation, reconstruction loss and cross product for state-action interaction. However, MultiDismantler aims at finding the cheapest way of destroying the LMCC in a one-to-one interdependent network, whereas FINDER aims at finding the cheapest way of destroying the largest connected component in a single-layer network. Being the methods intended to solve radically different dismantling problems, they differ in key aspects. For example, MultiDismantler relies on the GMM-based training versus the BA-based one of FINDER. Also, MultiDismantler relies on a MGNN, whereas FINDER takes advantage on a single-layer graph neural network. For a detailed discussion on differences and similarities between MultiDismantler and FINDER, see Supplementary Section 1.

Experimental setup

We train two MultiDismantler agents with identical training architectures but distinct reward functions to handle node dismantling tasks with unit and degree costs. Also, we consider cost values randomly

Table 1 | Real-world network robustness under random failure with different protection strategies

Dataset	HDA	CI	MinSum	FINDER	NIRM	CoreHLDA	EMD	MultiDismantler
USAir	0.421±0.032	0.366±0.064	0.377±0.046	0.437±0.039	0.419±0.024	0.402±0.054	0.402±0.038	0.419±0.024
FAO	0.410±0.007	0.408±0.006	0.406±0.004	0.408±0.003	0.413±0.009	0.410±0.006	0.408±0.006	0.411±0.007
Celegans	0.379±0.012	0.389±0.008	0.371±0.014	0.384±0.004	0.392±0.010	0.391±0.010	0.395±0.012	0.390±0.007
Drosophila	0.279±0.011	0.277±0.013	0.264±0.016	0.289±0.007	0.272±0.012	0.283±0.017	0.291±0.013	0.284±0.011
Fb&Tw	0.463±0.003	0.460±0.003	0.462±0.003	0.463±0.004	0.462±0.003	0.458±0.003	0.462±0.004	0.465±0.003
NetSci	0.325±0.006	0.318±0.007	0.315±0.013	0.324±0.005	0.317±0.010	0.322±0.009	0.317±0.012	0.327±0.009
Sacchpomb	0.237±0.001	0.235±0.003	0.218±0.006	0.243±0.002	0.239±0.004	0.243±0.003	0.246±0.001	0.246±0.002
Homo	0.211±0.001	0.207±0.001	0.185±0.012	0.213±0.002	0.203±0.002	0.213±0.001	0.214±0.002	0.217±0.001
Sanremo2016	0.207±0.001	0.202±0.001	0.138±0.005	0.204±0.002	0.198±0.001	0.209±0.005	0.210±0.001	0.212±0.001

The results are averaged over five realizations; errors denote the standard deviation over these realizations. The best results are highlighted with bold (corresponding to the highest AUDC values).

generated from uniform, normal and Poisson distributions. We perform tests on synthetic networks generated according to the GMM model. Also, we analyse nine real-world multilayer networks. We compare our results against various state-of-the-art dismantling methods, including HDA, CI, MinSum, FINDER and NIRM. These algorithms are primarily designed for the dismantling of single-layer networks. We apply them to each layer individually and then generate global scores by combining individual-layer scores in different ways (see ‘Other dismantling algorithms’ section in the Methods for details, and see Supplementary Section 3.6 for other cross-layer combination methods). Results are not very sensitive to the rule used to combine individual-layer scores into global ones, as already noted by Osat et al.¹⁹. Also, we compare MultiDismantler against the only existing dismantling algorithms specifically designed for interdependent networks, namely EMD²⁰ and CoreHLDA²². Both these algorithms are fundamentally different from MultiDismantler as they do not have a machine-learning component; instead, they leverage specifically tailored centrality metrics to dismantle interdependent networks.

Results on synthetic networks

Dismantling performance on synthetic networks. To demonstrate the universal advantages of our algorithm across various types of network, we vary the main parameters of the GMM model to generate several networks with different intralayer structures and different interlayer correlation levels. Figure 2a,b shows MultiDismantler’s superior dismantling performance on various synthetic networks with unit cost and degree cost. Supplementary Tables 10 and 11 highlight that our algorithm outperforms all baseline approaches over all synthetic networks: the AUDC and C values are respectively 5.9% and 2.6% lower than the second-best network dismantling algorithm EMD under unit cost, and 8.3% and 2.4% under degree cost.

Predicting the ground-truth optimal dismantling sequence. On very small networks, we evaluate all permutations to determine the optimal dismantling sequence with minimal AUDC in the unit-cost version of the problem. In this analysis, we consider the network fully dismantled when the LMCC contains only one node. Supplementary Table 6b uses the 1-normalized edit distance to quantify the similarity between the dismantling sequences identified by different algorithms and the optimal dismantling sequence computed via brute-force search. Supplementary Table 6a demonstrates that the dismantling sequence computed by MultiDismantler is identical to the optimal sequence and achieves the minimum dismantling AUDC values.

Results on real-world networks

Dismantling performance and inductive learning capacity on real-world networks. MultiDismantler is trained on a large number of synthetic networks and then applied to real-world networks.

We evaluate MultiDismantler on various real-world networks from different domains^{40–45}. Figure 2c,d shows that MultiDismantler on average outperforms all other methods in both the unit- and degree-cost problems. MultiDismantler’s cumulative AUDC is 8.4% lower than the second-best EMD algorithm for degree cost, and the cumulative C value is 3.0% lower than the second-best dismantling algorithm FINDER for unit cost. We refer the readers to Supplementary Tables 5 and 12 for results concerning dismantling experiments on real-world networks.

Figure 3 illustrates the process of dismantling two real multilayer networks using various methods with unit and degree cost. MultiDismantler consistently achieves the lowest LMCC for the same fraction of removal node costs compared with other dismantling methods on most real networks, indicating that MultiDismantler is more effective in identifying the dismantling solution (see Supplementary Fig. 4 for the dismantling process on other real-world networks under unit removal cost). In addition, for some networks, the MultiDismantler’s curve may initially be higher than those of other methods but then decline more rapidly. This suggests that the method predicts a long-term dismantling strategy: while some target nodes may not directly cause sizable network disruption, they are essential in rendering the network vulnerable, making it prone to rapid breakdown upon the removal of other nodes. To further assess MultiDismantler’s effectiveness on networks with uncertain node removal costs, we consider random costs extracted from the uniform, the normal and the Poisson distributions. Supplementary Table 7 shows that MultiDismantler exhibits excellent generalization ability, achieving the best performance on unseen normal and Poisson cost distributions, despite being trained on costs randomly extracted from the uniform distribution. This generalization capability of MultiDismantler demonstrates its practical value in real-world scenarios with uncertain node removal costs.

We apply MultiDismantler, trained on multilayer networks, to the single-layer network dismantling task. Supplementary Table 8 shows that MultiDismantler achieves the best dismantling performance. This can be attributed to the model’s ability to effectively leverage the structural features learned from multilayer networks (see Supplementary Section 3.5 for more details).

Analysing the key components of MultiDismantler. To determine the essential elements that contribute to the outstanding dismantling performance of MultiDismantler, we systematically alter its three main components under the unit-cost scenario: the attention mechanism in MGNN, the training corpus and the interlayer similarity in the GMM. Supplementary Table 4a validates the effectiveness of the interlayer attention. In the first column, we disable the interlayer message-passing mechanism, resulting in MultiDismantler’s inability to fuse representations from different layers, which leads to worse dismantling accuracy. Next, we replace the training corpus from the GMM with the BA model with randomly selected interdependent nodes. The first column in

Table 2 | Real-world network robustness under targeted attack with different protection strategies

Dataset	HDA	CI	MinSum	FINDER	NIRM	CoreHLDA	EMD	MultiDismantler
USAir	0.070	0.069	0.053	0.085	0.069	0.136	0.136	0.136
FAO	0.331	0.346	0.288	0.350	0.346	0.330	0.350	0.350
Celegans	0.226	0.226	0.161	0.240	0.240	0.239	0.239	0.239
Drosophila	0.117	0.112	0.057	0.119	0.117	0.123	0.127	0.132
Fb&Tw	0.326	0.326	0.332	0.330	0.338	0.332	0.343	0.348
NetSci	0.132	0.111	0.063	0.122	0.130	0.125	0.131	0.122
Sacchpomb	0.135	0.116	0.051	0.132	0.052	0.154	0.159	0.162
Homo	0.127	0.109	0.059	0.109	0.097	0.130	0.135	0.144
Sanremo2016	0.172	0.156	0.006	0.139	0.174	0.177	0.175	0.182

The best results are highlighted with bold (corresponding to the highest AUDC values).

Supplementary Table 4b indicates that MultiDismantler trained with the BA network performs much worse than the model trained with GMM. The performance of MultiDismantler degrades even more than for other algorithms, such as HDA and CI. Interlayer similarity also dramatically influences MultiDismantler's performance. We adjust the interlayer similarity correlation (g) of the GMM and train agents with varying degrees of interlayer similarity. As shown in Supplementary Table 4c, the performance is rather stable as g is varied and deteriorates only for extreme g values (that is, $g=0$ and $g=1$).

Application to the prevention of disease spreading and the attack–protect task in various real-world networks

Network connectivity is a crucial factor for network functionality, with the size of the LMCC being particularly relevant to problems such as optimal disease spreading⁴ and attack–protect strategies⁴⁶. Below, we provide evidence of the effectiveness of MultiDismantler in both applications.

Evaluating MultiDismantler's efficacy in immunization strategies. We leverage MultiDismantler to design immunization strategies for disease spreading on two social networks: Fb&Tw⁴⁷ and Sanremo2016⁴⁵. First, we compute the node dismantling sequence using MultiDismantler with unit cost, identifying nodes that play vital roles in network connectivity. We then determine the minimum number of nodes whose removal reduces the size of the LMCC to 1 and subsequently remove the top half of these nodes. For comparison, we remove the same number of nodes based on their importance rankings as identified by other algorithms. We run susceptible–infected–recovered (SIR) disease-spreading dynamics on the remaining multiplex network, setting all nodes to the susceptible state and randomly selecting one node as the seed for spreading. The SIR parameters are consistent with those used in the study by Patwardhan et al.³⁰: the spreading rate is set slightly higher than the critical one, that is, $\beta = \langle k \rangle / \langle k^2 \rangle + 0.02$, where $\langle k^m \rangle$ is the m th moment of the average degree distribution of the interdependent network. For simplicity, we use the same spreading rate for all layers.

Figure 4a,b demonstrates that an immunization strategy based on MultiDismantler effectively prevents the widespread outbreak of a disease. Compared with other algorithms, the immunization strategy informed by MultiDismantler dramatically reduces the number of individuals that must be immunized to suppress the spread of an epidemic.

MultiDismantler's performance in protecting network connectedness under targeted attacks and random failures. We protect the first 1% nodes in the sequence of removal identified by MultiDismantler or the other competing methods. We then perturb the network by removing either 1% of the remaining nodes at random (that is, random-failure protocol) or the top 1% nodes according to MultiDismantler that are

not in the protected set (that is, targeted attack). We then iterate the above node-protection and node-removal processes until the size of the LMCC reaches 1 or the number of attacked nodes exceeds the number of effective nodes in the residual network (that is, excluding already deleted and isolated nodes). It is worth noting that we select 1% of the initial network's nodes every iteration, rather than 1% of the residual network's nodes. In Tables 1 and 2, we summarize our results. MultiDismantler maintains network robustness, outperforming other algorithms under both the targeted-attack and the random-failure scenarios. The adaptability of MultiDismantler to both random and targeted failures underscores its versatility and reliability. By protecting the most critical nodes identified through its dismantling sequence, the algorithm not only mitigates immediate damage but also fortifies the network against subsequent failures. This dual capability of dismantling and protecting networks highlights MultiDismantler's comprehensive approach to network robustness, making it a valuable tool for enhancing the resilience of networks.

Figure 4c,d shows the dismantling process of the Homo network with different protection strategies. We observe that MultiDismantler can often achieve the highest LMCC for the same fraction of removal node costs, in cases of targeted attacks and random failures. This suggests that the model is highly effective in optimizing the network's robustness and stability by accurately identifying and protecting the nodes that play crucial roles in maintaining network connectedness.

Conclusion

In this study, we address the problem of dismantling one-to-one interdependent networks using a deep-learning-aided approach named MultiDismantler. The proposed method incorporates a multiplex embedding framework to encode networks and a DQN-based decoder to effectively train agents in learning complex dismantling strategies. Extensive experiments on both synthetic and real-world networks demonstrate the overall superior performance of MultiDismantler compared with existing state-of-the-art dismantling algorithms. Compared with other dismantling algorithms, MultiDismantler shows the highest similarity to the optimal network dismantling sequence computed via brute-force search. MultiDismantler exhibits remarkable generalization capabilities across various network topologies and node removal cost distributions, meaning that it can be safely applied to previously unseen networks. Furthermore, although MultiDismantler serves to approximate solutions to the network dismantling problem, it is useful also in designing solutions of related problems, such as the prevention of disease spreading in social networks and the attack-and-protect task in interdependent critical infrastructures.

Our work underscores the potential of combining graph neural networks with reinforcement learning for complex network optimization tasks and opens avenues for future research aimed at enhancing the scalability and adaptability of such models for broader applications.

One of these applications could be the extension of MultiDismantler to dismantling problems valid for other multilayer structures, for example, networks composed of many-to-many interdependent layers¹⁵ or networks characterized by redundant interdependencies⁴⁸. We leave such extensions to future research.

Methods

Dismantling multilayer interdependent networks

Problem formulation. We consider a network consisting of two layers of interactions. Nodes in the two layers are one-to-one interdependent. For simplicity of description but without loss of generality, we do not make a formal distinction between one node and its replica in the other layer; instead, we act as if the two layers share the same set of nodes. Thus, we denote the network as $G = (V, E^{(1)}, E^{(2)})$, where V is the set of nodes shared across the layers, with $|V| = N$ denoting the total number of nodes, and $E^{(\ell)}$ are the edges in layer $\ell = 1, 2$. The generic element of the adjacency matrix of layer ℓ is $A_{ij}^{(\ell)} = 1$ if nodes i and j are connected, or $A_{ij}^{(\ell)} = 0$ if they are not connected.

Large-scale connectedness of the network is quantified in terms of the fraction of nodes that belong to the LMCC of the network as $P_\infty = \frac{N_{\text{LMCC}}}{N_i}$, where N_{LMCC} is the number of nodes in the LMCC and N_i is the number of nodes in the initial LMCC, when all N nodes are present in the network¹⁶. The size of the LMCC can be reduced by removing nodes belonging to a subset $S \subseteq V$. Based on our definitions, if $S = \emptyset$, then $P_\infty(\emptyset) = 1$; if $S \neq \emptyset$, then $P_\infty(S) \leq 1$; also, $P_\infty(V) = 0$.

Network dismantling, often referred as the optimal percolation problem, can be seen as the constrained minimization problem

$$S^*(C) = \arg \min_{S | P_\infty(S)=C} P_\infty(S). \quad (1)$$

The constraint is imposed on the value of the cost function $F(S)$ of removing elements of the set S . In this Article, we consider three main types of cost function: (1) unit cost, (2) degree cost and (3) random cost. In the unit-cost version of the problem, the cost function associated to the set S equals its size, that is, $F(S) = |S|$. The degree-cost function of variant 2 is defined as $F(S) = \sum_{s \in S} \sum_{\ell=1}^2 k_s^{(\ell)} - \sum_{s,t \in S} A_{st}^{(\ell)}$, where $k_s^{(\ell)} = \sum_{j \in V} A_{sj}^{(\ell)}$ is the degree of node s in layer ℓ , the sums run over all nodes in the set S , and edges shared by nodes within the set S are counted only once. In the random-cost variant, the cost associated with each node s is $F(S) = \sum_{s \in S} \sum_{\ell=1}^2 u_s^{(\ell)}$, where $u_s^{(\ell)}$ is extracted at random from the uniform distribution defined in the interval $[0, 1]$, the normal distribution with average 0.5 and variance 0.1, or the Poisson distribution with average equal to 5.

An important aspect in the characterization of the optimization problem is the identification of the minimum-cost set of nodes able to lead to the disappearance of a macroscopic LMCC. Such a condition is defined in the problem

$$S_c = \arg \min_{S | P_\infty(S) \leq 1/\sqrt{N_i}} F(S). \quad (2)$$

Essentially, only sets S that can reduce P_∞ below the conventional threshold value $1/\sqrt{N_i}$ are considered as potential solutions to the problem³⁶.

Approximating solutions of the network dismantling problem. All algorithms considered in this Article construct approximate solutions to the dismantling problem sequentially, meaning that the set corresponding to the proposed solution is built by adding one element at a time. Indicate with $r_1, r_2, \dots, r_e, \dots, r_N$ the labels of the ranked nodes according to the algorithm at hand. Then, define $\tilde{S}_t = \bigcup_{e=1}^t \{r_e\}$, that is, the approximate solution of the algorithm when the set is composed of exactly t elements. By definition, $\tilde{S}_0 = \emptyset$ and $\tilde{S}_N = V$. We clearly have that $P_\infty(\tilde{S}_{t-1}) \geq P_\infty(\tilde{S}_t)$ and $F(\tilde{S}_t) \geq F(\tilde{S}_{t-1})$ for all $t = 1, \dots, N$.

We evaluate the performance of an approximate algorithm by measuring the AUDC as

$$\text{AUDC} = \frac{1}{F(V)} \sum_{t=1}^N P_\infty(\tilde{S}_t) [F(\tilde{S}_t) - F(\tilde{S}_{t-1})], \quad (3)$$

where $F(V)$ is the cost associated to the removal of all nodes from the graph. This is the generalization of the so-called robustness metric introduced by Schneider et al.⁴⁹. For computational reasons, we approximate AUDC by summing only the first N_c contributions such that $P_\infty(\tilde{S}_t) = 1/N_t$ for $t = 0, \dots, N_c$. This represents a good approximation of equation (3).

Also, we evaluate the performance of an approximate algorithm to solve the problem of equation (2) by measuring $F(\tilde{S}_c)$, where $\tilde{S}_c = \arg \min_{\tilde{S}_c | P_\infty(\tilde{S}_c) \leq 1/\sqrt{N_i}} F(\tilde{S}_c)$. To make the metric comparable across networks and/or variants of the dismantling problem, we define the dismantling cost as

$$C^* = \frac{F(\tilde{S}_c)}{F(V)}. \quad (4)$$

Note that both AUDC and C^* are defined in the interval $[0, 1]$. Also for both metrics, low values indicate good performance of the solution of the dismantling problem, whereas values close to 1 denote poor performance.

MultiDismantler

We provide below details on the encoder, decoder, loss function and dismantling strategy used in MultiDismantler.

Encoder. The framework of the MGNN is shown in Supplementary Fig. 5. MGNN consists of an intralayer graph convolutional neural network and an interlayer attention compute module to aggregate information from other layers.

Initializing node features in network embedding plays a crucial role in establishing a solid foundation and speeding up convergence. Drawing inspiration from the feature initialization method used in FINDER, MGNN utilizes a multilayer perceptron (MLP) to encode node feature

$$\mathbf{h}_v^{0,(\ell)} = \text{Norm} (\text{ReLU} (\mathbf{W}_1 \cdot \mathbf{x}_v^{(\ell)})), \quad (5)$$

where $\mathbf{x}_v^{(\ell)}$ is the normalized degree of node v in layer ℓ (dividing the degree of a node by the maximum degree in the layer), $\text{ReLU}(\cdot) = \max(0, \cdot)$ is the nonlinear activation function, \mathbf{W}_1 is a trainable weight matrix and $\text{Norm}(\cdot)$ denotes the L2 normalization.

After obtaining the initialized features, we apply the intralayer graph convolutional neural network, which iteratively updates the node embeddings by aggregating information from the node's previous features and the representations of its neighbours.

$$\mathbf{h}_v^{k,(\ell)} \leftarrow \text{Norm} \left(\text{ReLU} \left(\mathbf{W}_4 \cdot \left(\left\{ \mathbf{W}_3 \cdot \mathbf{h}_v^{k-1,(\ell)} \right\} \cup \left\{ \mathbf{W}_2 \cdot \sum_{j \in N(v)^{(\ell)}} \mathbf{h}_j^{k-1,(\ell)} \right\} \right) \right) \right). \quad (6)$$

In equation (6), $\mathbf{h}_v^{k,(\ell)}$ represents the learned embedding of node v in layer ℓ for the k th forward propagation step, which captures k hops neighbours' information, \mathbf{W}_2 , \mathbf{W}_3 and \mathbf{W}_4 are trainable weight matrices, \cup denotes the concatenation operation and $N(v)^{(\ell)}$ denotes the set of neighbours of node v in layer ℓ . The final intralayer node representation can be represented as $\mathbf{h}_v^{(\ell)}$.

In multilayer interdependent networks, nodes across different layers depend one on another. Given the heterogeneity of interlayer interaction and to make the cross-layer influence comparable, we first apply a mapping function to project node representations from different layers into a shared latent space, which is formulated as

$$\mathbf{h}_v^{(\ell)} = \tanh(\mathbf{W}_5 \cdot \mathbf{h}_v^{(\ell)} + \mathbf{b}_1), \quad (7)$$

where \mathbf{W}_5 is a trainable weight matrix, \mathbf{b}_1 denotes a trainable bias vector and \tanh is the nonlinear activation function.

Previous aggregation algorithms typically use fixed cross-layer weights to quantify interlayer influence, overlooking the variability of interlayer connection weights even within the same layer. To capture these high-order cross-layer node interaction relations and enhance the encoding expressiveness of our MGNN architecture, we use a node-level interlayer attention mechanism that captures information from across layers, enabling the model to effectively learn interactions between different layers of the interdependent network:

$$\alpha_v^{(\ell \leftarrow q)} = \frac{e^{\mathbf{W}_6 \cdot (\mathbf{h}_v^{(\ell)} \otimes \mathbf{h}_v^{(q)}) + \mathbf{b}_2}}{\sum_{p=1}^P e^{\mathbf{W}_6 \cdot (\mathbf{h}_v^{(\ell)} \otimes \mathbf{h}_v^{(p)}) + \mathbf{b}_2}}, \quad (8)$$

where $\alpha_v^{(\ell \leftarrow q)}$ is the computed interlayer attention of node v from layer q to ℓ , \mathbf{W}_6 is a trainable weight matrix, \mathbf{b}_2 denotes a bias vector, P is the layer number of the multiplex network and \otimes stands for the Hadamard product.

MGNN aggregates the learned intralayer representation and interlayer weights influence to form the final node embedding using

$$\mathbf{z}_v^{(\ell)} = \mathbf{h}_v^{(\ell)} + \sum_{p=1, p \neq \ell}^P \alpha_v^{(\ell \leftarrow p)} \cdot \mathbf{h}_v^{(p)}, \quad (9)$$

where $\mathbf{z}_v^{(\ell)}$ is the final representation of node v in layer ℓ .

Decoder. The decoder framework takes the network state and node action vector as input and computes Q values (the expected returns) for all possible actions with a neural network. In this Article, we apply node representations learned from MGNN as action vectors; $\mathbf{z}_v^{(\ell)}$ is the action vector of node v in layer ℓ . Network state represents the current network topology after the removal of nodes, considering networks are distributed systems, each node only has a partial view of the network, so we add a virtual node s to denote the state in each layer. In layer ℓ , all residual nodes are directed neighbours of s but node s is not a neighbour to any other node in that layer. This design helps to avoid over-smoothing and provides a comprehensive view of the network state. We use $\mathbf{z}_s^{(\ell)}$ to denote the embedding vector of the virtual node s in layer ℓ . We then utilize a two-layer MLP to model the expected cumulative layer-level reward (Q value) of an action under a given network state with

$$\mathbf{Q}^{(\ell)}(s, a_v) = \mathbf{M}_1 \cdot \text{ReLU}(\mathbf{P}_s^{(\ell)} \times \mathbf{z}_v^{(\ell)} \cdot \mathbf{M}_2), \quad (10)$$

where $\mathbf{Q}^{(\ell)}(s, a_v)$ denotes the Q -value action a_v , that is, removing the node v , under layer state s in layer ℓ , \mathbf{M}_1 and \mathbf{M}_2 are trainable weight matrices, and \times denotes the outer product operation that can model fine dependencies between states and actions¹⁴.

Nodes in different layers have various expected rewards, and $\mathbf{Q}^{(\ell)}(s, a_v)$ reflects only the partial expected reward of removing node v ; for example, node v might serve as a core node connecting different communities in one layer but act as a peripheral node in another layer. To have a comprehensive view of node importance from various layers, we use a layer-level fusion mechanism to quantify the final node removal return with

$$\omega^{(\ell)} = \frac{e^{\mathbf{M}_4 \cdot \text{ReLU}(\mathbf{M}_3 \cdot \mathbf{z}_s^{(\ell)})}}{\sum_{p=1}^P e^{\mathbf{M}_4 \cdot \text{ReLU}(\mathbf{M}_3 \cdot \mathbf{z}_s^{(p)})}}, \quad (11)$$

where $\omega^{(\ell)}$ denotes the reward importance of layer ℓ , and \mathbf{M}_3 and \mathbf{M}_4 are trainable matrices.

We aggregate the final expected return of node v across all layers in

$$\mathbf{Q}(s, a_v) = \sum_{p=1}^P \omega^{(p)} \cdot \mathbf{Q}^{(p)}(s, a_v), \quad (12)$$

where $\mathbf{Q}(s, a_v)$ denotes the expected Q value taking action a_v to remove node v under the state s , where s is the overall state of the interconnected network.

Loss function and dismantling process. For the observed reward, we use the impact of node removal on the real network's connectivity to quantify the direct reward of removal costs as

$$r(v) = P_\infty(\{v\})F(\{v\}), \quad (13)$$

where $P_\infty(\{v\})$ is the relative size of the LMCC of the graph when node v is removed, and $F(\{v\})$ is the cost of removing node v .

The impact of removing a node might not be immediately apparent and can have delayed effects on the network's connectivity. We apply n -step DQN to put n -step $(s_t, a_t, r_t, \dots, r_{t+n-1}, s_{t+n})$ in a replay buffer to better capture the delayed effects by considering the cumulative target reward over multiple steps. The target Q value is composed of the cumulative observed rewards, r_{t+n} , and the estimated maximum future reward from the next $t+n$ state under potential action a' is denoted as $\max_{a'} \hat{\mathbf{Q}}(s_{t+n}, a')$. The decoder framework evaluates an agent's predicted rewards of action a under state s in time t , denoted as $\mathbf{Q}(s_t, a_t)$. The Q -learning loss aims to minimize the difference between the predicted Q value and the target Q value

$$\text{Loss}_Q = \mathbb{E}_{(s_t, a_t, r_{t+n}, s_{t+n}) \sim U(B)} \left[\left(r_{t+n} + \rho \max_{a'} \hat{\mathbf{Q}}(s_{t+n}, a'; \hat{\Theta}_N) - \mathbf{Q}(s_t, a_t; \Theta_N) \right)^2 \right], \quad (14)$$

where $U(B)$ denotes the uniform sampling distribution over the experience replay buffer B , Θ_N is current network parameters and $\hat{\Theta}_N$ is the target network parameters which are updated periodically from Θ_N , and ρ is the discount factor that determines the importance of future rewards.

We also add a graph reconstruction loss⁵⁰ to preserve the network topology information with

$$\text{Loss}_R = \sum_{p=1}^P \sum_{i,j=1}^N A_{ij}^{(\ell)} \left\| \mathbf{z}_i^{(\ell)} - \mathbf{z}_j^{(\ell)} \right\|_2^2, \quad (15)$$

where $\|\cdot\|_2^2$ is the squared L2 norm.

The total loss Loss_T is composed of the Q -learning loss Loss_Q , and the graph reconstruction loss Loss_R

$$\text{Loss}_T = \text{Loss}_Q + \beta \text{Loss}_R, \quad (16)$$

where β is a hyperparameter that balances the importance of two losses.

Building on the previously established encoding, decoding and optimization framework, we use a greedy selection procedure to determine the optimal action and node to remove based on the Q value. During the training phase, an ϵ -greedy strategy is used, where the action with the highest Q value is selected with a probability of $(1 - \epsilon)$, while a random action is chosen with probability ϵ . Similar to FINDER¹⁴, the value of ϵ is gradually reduced from 1 to 0.05 over 10,000 episodes to balance exploration and exploitation. In the application phase, the well-trained agent model is used to compute Q values for each node. The Q -value sequence for state and action is represented as $(Q(s, a_1), Q(s, a_2), \dots, Q(s, a_m))$, where m is the number of remaining nodes. In the application phase, we always select the actions with the highest Q values to take. The node selection and removal process continues until the network is fully dismantled.

Other dismantling algorithms

Here, we briefly describe the other dismantling algorithms used in our experiments. For single-layer dismantling algorithms, we explain how they can be generalized and applied to multilayer networks. These generalized algorithms assign to each node a global score that is equal to the maximum of a suitably chosen centrality metric estimated on each individual layer. This heuristic recipe was introduced by ref. 28 as a generalization of the targeted-attack protocol from single-layer to one-to-one interdependent networks. We stress, however, that this is not necessarily the optimal way of aggregating single-layer metrics into a global one; future work could leverage a deep learning approach to infer the appropriate rule of score aggregation able to maximize the performance of the resulting algorithms.

HDA. We assign to each node i the score $s_i = \max(k_i^{(1)}, k_i^{(2)})$, and k_i is the degree of node i , which ranks nodes solely by degree. We remove node with the highest degree. In cases where nodes share identical degrees, we randomly select and remove one of these nodes. Further, once a node is removed from the network, the degrees of the remaining nodes are updated¹¹.

CI. We use the adaptive version of the CI centrality⁴. In each layer, the score assigned to each node i is a function of the number and degree of other nodes at distance ν from i . ν is a tunable parameter. For $\nu = 0$, the metric reduces to HDA. Here, we use $\nu = 1$. In our experiments, we assign to a given node a score equal to the maximum of its CI scores in the two layers. Furthermore, once a node is removed from the network, the scores of the remaining nodes are updated. The implementation of the CI algorithm we applied is available via GitHub at <https://github.com/zhfkt/ComplexCi>.

MinSum. We first apply the MinSum⁹ algorithm to each layer of the interconnected network to obtain their dismantling sequences. The position of node i in the dismantling sequence of layer ℓ is represented as $r_i^{(\ell)}$. We then assign each node i a score $s_i = \max(1/r_i^{(1)}, 1/r_i^{(2)})$ and remove node with the biggest score. For nodes with the same score, we randomly select and remove one of those nodes. The implementation of the MinSum algorithm we applied is available via GitHub at <https://github.com/abraunst/decycler>.

FINDER. FINDER uses a DRL framework to identify critical nodes for single-layer networks¹⁴. FINDER estimates the scores $q_i^{(\ell)}$ representing the expected return if node i is removed from layer ℓ . To apply FINDER in multilayer interdependent networks, we assign to each node i the score $s_i = \max(q_i^{(1)}, q_i^{(2)})$ for all i and remove the node with the largest score. We utilized the well-trained FINDER version in the unit-cost scenario. Such a FINDER implementation is available via GitHub at <https://github.com/FFranky/FINDER>. As for other weighted scenarios such as degree and random removal costs, we retrained the FINDER model with the same training iterations with Multidismantler for a fair comparison.

NIRM. The NIRM uses both local and global scoring mechanisms to learn the final scores for each node in a network layer. The final score of node i is given by $s_i = \max(m_i^{(1)}, m_i^{(2)})$, where $m_i^{(\ell)}$ denotes the score of node i in layer ℓ . For nodes with identical final scores, we randomly select and remove one of those nodes. The implementation of the NIRM algorithm we applied is available via GitHub at <https://github.com/JiazhengZhang/NIRM>.

EMD. The EMD²⁰ is specifically designed for dismantling interdependent networks. The weight of a node is determined by the sum of the impact values of its neighbours, with the impact value depending on both the weight and the degree of each neighbouring node.

CoreHLDA. CoreHLDA²² consists of network recycling, tree breaking and refining processes to dismantle multiplex networks. It applies a novel score fusing method in network recycling and tree breaking to combine the dismantling scores from the single-layer network.

Reporting summary

Further information on research design is available in the Nature Portfolio Reporting Summary linked to this article.

Data availability

All data used in this Article are publicly available. To facilitate the reproducibility of our results, all necessary data are available at <https://doi.org/10.24433/CO.0638082.v1> (ref. 51).

Code availability

The code developed for this research is available at <https://doi.org/10.24433/CO.0638082.v1> (ref. 51).

References

- Watts, D. J. & Strogatz, S. H. Collective dynamics of ‘small-world’ networks. *Nature* **393**, 440–442 (1998).
- Hwang, D.-U. Complex networks: structure and dynamics. *Phys. Rep.* **424**, 175–308 (2006).
- Cohen, R., Erez, K., ben-Avraham, D. & Havlin, S. Breakdown of the internet under intentional attack. *Phys. Rev. Lett.* **86**, 3682 (2001).
- Morone, F. & Makse, H. A. Influence maximization in complex networks through optimal percolation. *Nature* **524**, 65–68 (2015).
- Barabási, A.-L., Gulbahce, N. & Loscalzo, J. Network medicine: a network-based approach to human disease. *Nat. Rev. Genet.* **12**, 56–68 (2011).
- Carreras, B. A., Lynch, V. E., Dobson, I. & Newman, D. E. Critical points and transitions in an electric power transmission model for cascading failure blackouts. *Chaos* **12**, 985–994 (2002).
- Bertagnoli, G., Gallotti, R. & De Domenico, M. Quantifying efficient information exchange in real network flows. *Commun. Phys.* **4**, 125 (2021).
- Chen, Y., Paul, G., Havlin, S., Liljeros, F. & Stanley, H. E. Finding a better immunization strategy. *Phys. Rev. Lett.* **101**, 058701 (2008).
- Braunstein, A., Dall’Asta, L., Semerjian, G. & Zdeborová, L. Network dismantling. *Proc. Natl Acad. Sci. USA* **113**, 12368–12373 (2016).
- Artimo, O. et al. Robustness and resilience of complex networks. *Nat. Rev. Phys.* **6**, 114–131 (2024).
- Chen, W., Wang, Y. & Yang, S. Efficient influence maximization in social networks. In *Proc. 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 199–208 (2009).
- Grassia, M., De Domenico, M. & Mangioni, G. Machine learning dismantling and early-warning signals of disintegration in complex systems. *Nat. Commun.* **12**, 5190 (2021).
- Zhang, J. & Wang, B. Dismantling complex networks by a neural model trained from tiny networks. In *Proc. 31st ACM International Conference on Information & Knowledge Management* 2559–2568 (2022).
- Fan, C., Zeng, L., Sun, Y. & Liu, Y.-Y. Finding key players in complex networks through deep reinforcement learning. *Nat. Mach. Intell.* **2**, 317–324 (2020).
- Bianconi, G. *Multilayer Networks: Structure and Function* (Oxford Univ. Press, 2018).
- Buldyrev, S. V., Parshani, R., Paul, G., Stanley, H. E. & Havlin, S. Catastrophic cascade of failures in interdependent networks. *Nature* **464**, 1025–1028 (2010).
- Vespignani, A. The fragility of interdependency. *Nature* **464**, 984–985 (2010).

18. Chen, Y., Liu, Y., Tang, M. & Lai, Y.-C. Epidemic dynamics with non-Markovian travel in multilayer networks. *Commun. Phys.* **6**, 263 (2023).
19. Osat, S., Faqeeh, A. & Radicchi, F. Optimal percolation on multiplex networks. *Nat. Commun.* **8**, 1540 (2017).
20. Baxter, G. J., Timár, G. & Mendes, J. Targeted damage to interdependent networks. *Phys. Rev. E* **98**, 032307 (2018).
21. Coghi, F., Radicchi, F. & Bianconi, G. Controlling the uncertain response of real multiplex networks to random damage. *Phys. Rev. E* **98**, 062317 (2018).
22. Han, J., Tang, S., Shi, Y., Zhao, L. & Li, J. An efficient layer node attack strategy to dismantle large multiplex networks. *Eur. Phys. J. B* **94**, 74 (2021).
23. Schuetz, M. J., Brubaker, J. K. & Katzgraber, H. G. Combinatorial optimization with physics-inspired graph neural networks. *Nat. Mach. Intell.* **4**, 367–377 (2022).
24. Khalil, E., Dai, H., Zhang, Y., Dilkin, B. & Song, L. Learning combinatorial optimization algorithms over graphs. *Adv. Neural Inf. Process. Syst.* **30**, (2017).
25. Wu, C. P., Lou, Y., Li, W., Xie, S. L. & Chen, G. R. A multitask network robustness analysis system based on the graph isomorphism network. *IEEE Trans. Cybernetics* **54**, 6630–6642 (2024).
26. Kleineberg, K.-K., Boguná, M., Ángeles Serrano, M. & Papadopoulos, F. Hidden geometric correlations in real multiplex networks. *Nat. Phys.* **12**, 1076–1081 (2016).
27. Boguna, M. et al. Network geometry. *Nat. Rev. Phys.* **3**, 114–135 (2021).
28. Kleineberg, K.-K., Buzna, L., Papadopoulos, F., Boguñá, M. & Serrano, M. Á. Geometric correlations mitigate the extreme vulnerability of multiplex networks against targeted attacks. *Phys. Rev. Lett.* **118**, 218301 (2017).
29. Faqeeh, A., Osat, S. & Radicchi, F. Characterizing the analogy between hyperbolic embedding and community structure of complex networks. *Phys. Rev. Lett.* **121**, 098301 (2018).
30. Patwardhan, S., Rao, V. K., Fortunato, S. & Radicchi, F. Epidemic spreading in group-structured populations. *Phys. Rev. X* **13**, 041054 (2023).
31. Sutton, R. S. Learning to predict by the methods of temporal differences. *Mach. Learn.* **3**, 9–44 (1988).
32. Ren, X.-L., Gleinig, N., Helbing, D. & Antulov-Fantulin, N. Generalized network dismantling. *Proc. Natl Acad. Sci. USA* **116**, 6554–6559 (2019).
33. Ramsey, F. P. in *Classic Papers in Combinatorics* 1–24 (Springer, 1987).
34. Kivelä, M. et al. Multilayer networks. *J. Complex Netw.* **2**, 203–271 (2014).
35. Lou, Y., Wang, L. & Guanrong, C. Structural robustness of complex networks: a survey of a posteriori measures. *IEEE Circuits Syst. Mag.* **23**, 12–35 (2017).
36. Clusella, P., Grassberger, P., Pérez-Reche, F. J. & Politi, A. Immunization and targeted destruction of networks using explosive percolation. *Phys. Rev. Lett.* **117**, 208301 (2016).
37. Ma, Y., Wang, S., Aggarwal, C. C., Yin, D. & Tang, J. Multi-dimensional graph convolutional networks. In *Proc. 2019 Siam International Conference on Data Mining* 657–665 (2019).
38. Dong, Y., Chawla, N. V. & Swami, A. metapath2vec: scalable representation learning for heterogeneous networks. In *Proc. 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 135–144 (2017).
39. Xu, S. et al. Topic-aware heterogeneous graph neural network for link prediction. In *Proc. 30th ACM International Conference on Information & Knowledge Management* 2261–2270 (2021).
40. Radicchi, F. Percolation in real interdependent networks. *Nat. Phys.* **11**, 597–602 (2015).
41. De Domenico, M., Nicosia, V., Arenas, A. & Latora, V. Structural reducibility of multilayer networks. *Nat. Commun.* **6**, 6864 (2015).
42. Stark, C. et al. Biogrid: a general repository for interaction datasets. *Nucleic Acids Res.* **34**, 535–539 (2006).
43. Cao, X. & Yu, Y. Bass: a bootstrapping approach for aligning heterogenous social networks. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Part I* 459–475 (2016).
44. De Domenico, M., Lancichinetti, A., Arenas, A. & Rosvall, M. Identifying modular flows on multilayer networks reveals highly overlapping organization in interconnected systems. *Phys. Rev. X* **5**, 011027 (2015).
45. De Domenico, M. & Altmann, E. G. Unraveling the origin of social bursts in collective attention. *Sci. Rep.* **10**, 4629 (2020).
46. Albert, R., Jeong, H. & Barabási, A.-L. Error and attack tolerance of complex networks. *Nature* **406**, 378–382 (2000).
47. Celli, F., Lascio, F. M. L. D., Magnani, M., Pacelli, B. & Rossi, L. Social network data and practices: the case of Friendfeed. In *International Conference on Social Computing, Behavioral Modeling and Prediction* 346–353 (Springer, 2010).
48. Radicchi, F. & Bianconi, G. Redundant interdependencies boost the robustness of multiplex networks. *Phys. Rev. X* **7**, 011013 (2017).
49. Schneider, C. M., Moreira, A. A., Andrade Jr, J. S., Havlin, S. & Herrmann, H. J. Mitigation of malicious attacks on networks. *Proc. Natl Acad. Sci. USA* **108**, 3838–3841 (2011).
50. Wang, D., Cui, P. & Zhu, W. Structural deep network embedding. In *Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 1225–1234 (2016).
51. Gu, W., Chen, Y., Li, L. & Hou, J. Deep-learning-aided dismantling of interdependent networks. *Code Ocean* <https://doi.org/10.24433/CO.0638082.v1> (2025).

Acknowledgements

This work was supported by grants from National Natural Science Foundation of China (grant numbers 42450183 and 72371014) and support from the Beijing University of Chemical Technology, grant number 11170044127, PY2514; partial support was also received from the Air Force Office of Scientific Research (grant numbers FA9550-21-1-0446 and FA9550-24-1-0039). The funders had no role in the study design, data collection, and analysis, the decision to publish or any opinions, findings, conclusions or recommendations expressed in the Article.

Author contributions

W.W. and F.R. wrote the paper. W.W., C.Y. and F.R. designed the model and experiments. W.W., C.Y. and L.L. performed the experiments. C.Y., L.L. and J.H. plotted the figures.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at <https://doi.org/10.1038/s42256-025-01070-2>.

Correspondence and requests for materials should be addressed to Chen Yang or Filippo Radicchi.

Peer review information *Nature Machine Intelligence* thanks Roger Guimerà and the other, anonymous, reviewer(s) for their contribution to the peer review of this work.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with

the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

© The Author(s), under exclusive licence to Springer Nature Limited 2025

Corresponding author(s): Filippo Radicchi

Last updated by author(s): 05/15/2025

Reporting Summary

Nature Portfolio wishes to improve the reproducibility of the work that we publish. This form provides structure for consistency and transparency in reporting. For further information on Nature Portfolio policies, see our [Editorial Policies](#) and the [Editorial Policy Checklist](#).

Statistics

For all statistical analyses, confirm that the following items are present in the figure legend, table legend, main text, or Methods section.

n/a	Confirmed
<input checked="" type="checkbox"/>	<input type="checkbox"/> The exact sample size (n) for each experimental group/condition, given as a discrete number and unit of measurement
<input checked="" type="checkbox"/>	<input type="checkbox"/> A statement on whether measurements were taken from distinct samples or whether the same sample was measured repeatedly
<input checked="" type="checkbox"/>	<input type="checkbox"/> The statistical test(s) used AND whether they are one- or two-sided <i>Only common tests should be described solely by name; describe more complex techniques in the Methods section.</i>
<input checked="" type="checkbox"/>	<input type="checkbox"/> A description of all covariates tested
<input checked="" type="checkbox"/>	<input type="checkbox"/> A description of any assumptions or corrections, such as tests of normality and adjustment for multiple comparisons
<input checked="" type="checkbox"/>	<input type="checkbox"/> A full description of the statistical parameters including central tendency (e.g. means) or other basic estimates (e.g. regression coefficient) AND variation (e.g. standard deviation) or associated estimates of uncertainty (e.g. confidence intervals)
<input checked="" type="checkbox"/>	<input type="checkbox"/> For null hypothesis testing, the test statistic (e.g. F , t , r) with confidence intervals, effect sizes, degrees of freedom and P value noted <i>Give P values as exact values whenever suitable.</i>
<input checked="" type="checkbox"/>	<input type="checkbox"/> For Bayesian analysis, information on the choice of priors and Markov chain Monte Carlo settings
<input checked="" type="checkbox"/>	<input type="checkbox"/> For hierarchical and complex designs, identification of the appropriate level for tests and full reporting of outcomes
<input checked="" type="checkbox"/>	<input type="checkbox"/> Estimates of effect sizes (e.g. Cohen's d , Pearson's r), indicating how they were calculated

Our web collection on [statistics for biologists](#) contains articles on many of the points above.

Software and code

Policy information about [availability of computer code](#)

Data collection Scripts for data processing were written in python(3.8), numpy(1.17.3), networkx(2.5.1)

Data analysis The training and evaluation framework used were implemented using python(3.8),pytorch(1.12.0),networkx(2.5.1), and others.

For manuscripts utilizing custom algorithms or software that are central to the research but not yet described in published literature, software must be made available to editors and reviewers. We strongly encourage code deposition in a community repository (e.g. GitHub). See the Nature Portfolio [guidelines for submitting code & software](#) for further information.

Data

Policy information about [availability of data](#)

All manuscripts must include a [data availability statement](#). This statement should provide the following information, where applicable:

- Accession codes, unique identifiers, or web links for publicly available datasets
- A description of any restrictions on data availability
- For clinical datasets or third party data, please ensure that the statement adheres to our [policy](#)

<https://codeocean.com/capsule/6460456>

Research involving human participants, their data, or biological material

Policy information about studies with [human participants or human data](#). See also policy information about [sex, gender \(identity/presentation\), and sexual orientation](#) and [race, ethnicity and racism](#).

Reporting on sex and gender	<input type="checkbox"/> no sex and gender
Reporting on race, ethnicity, or other socially relevant groupings	<input type="checkbox"/> no race, ethnicity or other socially relevant groupings
Population characteristics	<input type="checkbox"/> no population characteristics
Recruitment	<input type="checkbox"/> no recruitment participants
Ethics oversight	<input type="checkbox"/> no ethics oversight

Note that full information on the approval of the study protocol must also be provided in the manuscript.

Field-specific reporting

Please select the one below that is the best fit for your research. If you are not sure, read the appropriate sections before making your selection.

Life sciences Behavioural & social sciences Ecological, evolutionary & environmental sciences

For a reference copy of the document with all sections, see nature.com/documents/nr-reporting-summary-flat.pdf

Behavioural & social sciences study design

All studies must disclose on these points even when the disclosure is negative.

Study description	Identifying key nodes in the network
Research sample	Publicly available dataset
Sampling strategy	Publicly available dataset
Data collection	Publicly available dataset
Timing	Publicly available dataset
Data exclusions	no data exclusions
Non-participation	no participation
Randomization	no participation

Reporting for specific materials, systems and methods

We require information from authors about some types of materials, experimental systems and methods used in many studies. Here, indicate whether each material, system or method listed is relevant to your study. If you are not sure if a list item applies to your research, read the appropriate section before selecting a response.

Materials & experimental systems		Methods	
n/a	Involved in the study	n/a	Involved in the study
<input checked="" type="checkbox"/>	<input type="checkbox"/> Antibodies	<input checked="" type="checkbox"/>	<input type="checkbox"/> ChIP-seq
<input checked="" type="checkbox"/>	<input type="checkbox"/> Eukaryotic cell lines	<input checked="" type="checkbox"/>	<input type="checkbox"/> Flow cytometry
<input checked="" type="checkbox"/>	<input type="checkbox"/> Palaeontology and archaeology	<input checked="" type="checkbox"/>	<input type="checkbox"/> MRI-based neuroimaging
<input checked="" type="checkbox"/>	<input type="checkbox"/> Animals and other organisms		
<input checked="" type="checkbox"/>	<input type="checkbox"/> Clinical data		
<input checked="" type="checkbox"/>	<input type="checkbox"/> Dual use research of concern		
<input checked="" type="checkbox"/>	<input type="checkbox"/> Plants		

Plants

Seed stocks

no seed stocks

Novel plant genotypes

no novel plant genotypes

Authentication

no authentication