Latent Geometry-Driven Network Automata for Complex Network Dismantling

Marco Grassia †, Thomas Adler †, Ziheng Liao, Giuseppe Mangioni *, Carlo V. Cannistraci *

Posted Date: 6 November 2025

doi: 10.20944/preprints202505.2276.v2

Keywords: network robustness; network dismantling; network geometry; network science; complex systems; network automata; graphs; network topology

 doi:10.20944/preprints202505.2276.v2

Article

Latent Geometry-Driven Network Automata for Complex Network Dismantling

Marco Grassia [1], Thomas Adler [2,3], Ziheng Liao[2,3], Giuseppe Mangioni [1,*] and Carlo V. Cannistraci [2,3,4,*]

[1] Department of Electric, Electronic and Computer Engineering, University of Catania
[2] Center for Complex Network Intelligence (CCNI), Tsinghua Laboratory of Brain and Intelligence (THBI), Department of Psychological and Cognitive Sciences
[3] Department of Computer Science and Technology, Tsinghua University
[4] Department of Biomedical Engineering, Tsinghua University
* Correspondence: giuseppe.mangioni@unict.it (G.M.); kalokagathos.agon@gmail.com (C.V.C.)

Abstract

Complex networks model the structure and function of critical technological, biological, and communication systems. Network dismantling, the targeted removal of nodes to fragment a network, is essential for analyzing and improving system robustness. Existing dismantling methods suffer from key limitations: they depend on global structural knowledge, exhibit slow running times on large networks, and overlook the network's latent geometry, a key feature known to govern the dynamics of complex systems. Motivated by these findings, we introduce Latent Geometry-Driven Network Automata (LGD-NA), a novel framework that leverages local network automata rules to approximate effective link distances between interacting nodes. LGD-NA is able to identify critical nodes and capture latent manifold information of a network for effective and efficient dismantling. We show that this latent geometry-driven approach outperforms all existing dismantling algorithms, including machine learning methods such as graph neural networks. We also find that a simple common-neighbor-based network automata rule achieves near state-of-the-art performance, highlighting the effectiveness of minimal local information for dismantling. LGD-NA is extensively validated on the largest and most diverse collection of real-world networks to date (1,475 real-world networks across 32 complex systems domains) and scales efficiently to large networks via GPU acceleration. Finally, we leverage the explainability of our common-neighbor approach to engineer network robustness, substantially increasing the resilience of real-world networks. Our results confirm latent geometry as a fundamental principle for understanding the robustness of real-world systems, adding dismantling to the growing set of processes that network geometry can explain.

Keywords: network robustness; network dismantling; network geometry; network science; complex systems; network automata; graphs; network topology

1. Introduction

Complex networks are the backbone of our modern world, from the biological pathways within a cell to global financial and transportation systems [5]. While the interconnected nature of these systems is often a source of efficiency and strength, it also introduces profound vulnerabilities. A localized failure can be absorbed, or it can trigger a cascade of disruptions leading to a systemic collapse. Understanding this fragility is crucial, as the consequences are far-reaching: targeted disruptions can compromise cellular function in metabolic networks, dictate the spread of a virus through a social fabric, or cause catastrophic blackouts in power grids and failures in financial markets [6,7]. The formal study of these vulnerabilities is known as network dismantling. It addresses a fundamental question: what is the most efficient way to fragment a network by removing a minimal set of nodes or links, to disrupt its structural integrity and functional capacity? Answering this question is essential not only

 doi:10.20944/preprints202505.2276.v2

for predicting the impact of malicious attacks but, more importantly, for designing robust and resilient systems that can withstand them.

Efficient network dismantling is challenging because identifying the minimal set of nodes for optimal disruption is an NP-hard problem: no known algorithm can solve it efficiently for large networks [6]. This difficulty arises not only from the prohibitively large solution space but also from the structural complexity of real-world networks, which exhibit heterogeneous, fat-tailed connectivity [8–11] and intricate organizations such as modular and community structures [12], hierarchies [13,14], higher-order structures [15,16], and a latent geometry [1,17–20].

Node Betweenness Centrality (NBC) is a network centrality measure [21] that quantifies the importance of a node in terms of the fraction of the shortest paths that pass through it. NBC-based attack, where nodes are removed in order of their betweenness centrality, is considered one of, if not the best, method for network dismantling [22–25]. However, like many other dismantling techniques, it requires global knowledge of the entire network topology, and its high computational cost limits its scalability to large networks. These limitations are shared by many other state-of-the-art dismantling methods, which additionally rely on black-box machine learning models, and are rarely validated across large, diverse sets of real-world networks (see Tables 1, A3, and A2).

Latent geometry has been recognized as a key principle for understanding the structure and complexity of real-world networks. It explains essential features such as small-worldness, degree heterogeneity, clustering, and navigability, and drives critical processes like efficient information flow [17–20,26–29]

Work by Muscoloni et al. [1] revealed that betweenness centrality is a global latent geometry estimator: it approximates node distances in an underlying geometric space. They also introduced Repulsion-Attraction network automata rule 2 (RA2), a local latent geometry estimator that uses only first-neighbor connectivity. RA2 performed comparably to NBC in tasks such as network embedding and community detection, despite relying solely on local information.

This raises the first question: can latent geometry, whether estimated globally or locally, guide effective network dismantling? If complex systems run on a latent manifold, estimating it may offer a more efficient way to disrupt connectivity.

The second question concerns efficiency. While both NBC and RA2 have $O(Nm)$ complexity, where N is the number of nodes and m the number of links in a network, RA2 is significantly faster in practice because its local computations avoid NBC's large computational overhead. This motivates exploring whether local latent geometry estimators can match the dismantling performance of global methods like NBC while offering lower practical running time.

Motivated by these questions, we introduce the Latent Geometry-Driven Network Automata (LGD-NA) framework. Our first and primary contribution is the principle of (1) Latent Geometry-Driven (LGD) dismantling, where methods estimate effective node distances on a network's latent manifold to expose critical structural information. Specifically, our (2) LGD-NA framework uses local network automata rules to approximate these geometric distances; a node's summed distance to its neighbors estimates how critical it is for dismantling. Within this framework, we discovered that a (3) simple common neighbors-based rule, which we term Common Neighbor Dissimilarity (CND), is highly effective, achieving performance close to the state-of-the-art method, NBC. We prove the effectiveness of our approach through (4) comprehensive experimental validation on an ATLAS of 1,475 real-world networks across 32 complex systems domains, the largest and most diverse collection to date, showing that LGD-NA consistently outperforms all other existing dismantling algorithms, including machine learning methods. To enable dismantling at large scales, we implement (5) GPU-acceleration for LGD-NA, yielding remarkable running time advantages over methods like NBC. Finally, using the explainability of our CND measure, we introduce a new method for (6) engineering network robustness, substantially reducing the effectiveness of the best dismantling methods.

 doi:10.20944/preprints202505.2276.v2

Table 1. Number of real-world networks tested by dismantling algorithms, see Table A6 for more information.

Algorithm	Year	Networks	Ref.
Collective Influence (CI)	2016	2	Morone et al. [3]
CoreHD	2016	12	Zdeborová et al. [30]
Explosive Immunization (EI)	2016	5	Clusella et al. [31]
Min-Sum (MS)	2016	2	Braunstein et al. [2]
GND	2019	10	Ren et al. [32]
Resilience Centrality	2020	4	Zhang et al. [33]
GDM	2021	57	Grassia et al. [34]
CoreGDM	2023	15	Grassia and Mangioni [35]
Domirank Centrality	2024	6	Engsig et al. [22]
Fitness Centrality	2025	5	Servedio et al. [23]
LGD-NA	**2025**	**1,475**	**Ours**

2. Related Work

Latent Geometry of Complex Networks.

Many real-world networks are shaped by latent geometric manifolds of the complex systems that govern their topology and dynamics. These hidden geometries explain essential structural features such as small-worldness, degree heterogeneity, clustering, and community structure [1,17, 18,20,29,36,37]. The underlying metric space is not only descriptive but functional: it facilitates efficient routing and navigation with limited global knowledge [19,26–28]. Such properties emerge consistently across diverse systems, including biological, social, technological, and socio-ecological networks [17,18]. Latent geometries also enable predictive modeling of dynamical processes such as network growth [29,37,38], and epidemic spreading [39].

Latent Geometry Estimators.

Latent geometry estimators assign edge weights to approximate linked nodes' pairwise distances in the hidden geometric manifold. Among them, network automata rules based on the Repulsion-Attraction (RA) criterion use only local topological information to infer proximity in the latent space [1]. RA is grounded in the theory of network navigability [26], which posits that nodes with many non-overlapping neighbors tend to occupy distant regions in the latent space. Edges between such nodes receive higher dissimilarity scores due to strong repulsion, while those with many common neighbors are scored lower due to attraction.

RA1 and RA2 are network automata rules for approximating linked nodes' pairwise distances on the latent manifold of a complex network. These rules are categorized as network automata because they adopt only local information to infer the score of a link in the network without the need for pre-training of the rule. Note that RA1 and RA2 are **predictive network automata** that differ from generative network automata, which are rules created to generate artificial networks [8,37,38]. They were introduced to serve as pre-weighting strategies for approximating angular distances associated with node similarities in hyperbolic network embeddings. RA2 performed slightly better than RA1, so for this reason we will only consider RA2 in this study. RA2 defines dissimilarity between nodes i and j as:

$$\mathrm{RA2}(i,j) = \frac{1 + e_i + e_j + e_i \cdot e_j}{1 + \mathrm{CN}_{ij}}.$$

 doi:10.20944/preprints202505.2276.v2

where CN_{ij} is the number of common neighbors of nodes i and j, and e_i and e_j are the external degrees of i and j, representing the count of neighbors of i and j that are not involved in the common neighbors interactions.

In the same work, Muscoloni et al. also showed that betweenness centrality is a global latent geometry estimator. By comparing it with RA2, they demonstrated that both global (betweenness centrality) and local (RA) estimators can effectively capture latent geometry, achieving strong results in network embedding and community detection. See Table A1 for a comparison of estimators and Figure A1 for illustrative examples. See also Figure A9, showing how the RA2 measure indeed gives higher weights to nodes that are more central in the hyperbolic nPSO network, showing that these RA measures can effectively approximate latent geometric distances, in this case, in the hyperbolic space.

Topological centrality measures.

Degree, betweenness centrality, and their variants have all been used in the majority of dismantling studies [6], with betweenness centrality having been found to be the most effective strategy when applying dynamic dismantling, meaning the scores are recomputed after every step. Degree centrality ranks nodes by their number of neighbors, and betweenness centrality [21] counts how frequently a node lies on shortest paths. Other centrality variants include eigenvector centrality [40], which gives higher scores to nodes connected to other influential nodes. PageRank [41], based on a random walk model, favors nodes that receive many and high-quality links. Beyond these classical measures, several centrality indices have been developed specifically to capture aspects of network resilience. Fitness centrality [23], adapted from economic complexity theory, evaluates node importance through the capabilities of neighbors while penalizing connections to weak nodes. DomiRank [22] centrality models a competitive dynamic in which nodes gain or lose dominance, or importance, based on the relative strength of their neighbors. Resilience centrality [33], derived from a dynamical systems reduction, quantifies how a node's removal alters the system's resilience. See Table A2 for more information.

Statistical and Machine Learning Network Dismantling.

We focus on network dismantling for targeted attacks, where the goal is to fragment a network as efficiently as possible by removing selected nodes. Message passing-based methods such as Belief Propagation-guided Decimation (BPD) [4] and Min-Sum (MS) [2] use message-passing algorithms to decycle the network and then fragment the resulting forest with a tree-breaker algorithm, while CoreHD [30] achieves decycling by iteratively removing the highest-degree nodes from the 2-core of the network and also includes a tree-breaker algorithm. Decycling and dismantling are, in fact, closely related tasks, as a tree (or a forest) can be dismantled almost optimally [2]. Generalized Network Dismantling (GND) [32] targets nodes that maximize an approximated spectral partitioning. Collective Influence (CI) [3] targets nodes with maximal influence on their neighborhoods, and Explosive Immunization (EI) [31], uses explosive percolation dynamics. Machine learning-based methods include Graph Dismantling with Machine Learning (GDM) [34], which trains graph neural networks to predict optimal attack strategies in a supervised manner. FINDER [42] uses reinforcement learning instead to autonomously learn dismantling strategies without needing labeled data. CoreGDM [35] combines ideas from CoreHD and GDM as it attacks the 2-core of the network but uses machine learning models trained on optimal dismantling solutions to guide node removal. See Table A3 for more information.

3. Latent Geometry-Driven Network Automata

We introduce the Latent Geometry-Driven Network Automata (LGD-NA) framework. LGD-NA adopts a network automaton rule, such as RA2, to estimate latent geometric linked node pairwise distances and to assign edge weights based on these geometric distances. Then, it computes for each node its network centrality as a sum of the weights of adjacent edges. The higher this sum, the more a node dominates numerous and far-apart regions of the network, becoming a prioritized candidate for a targeted attack in the network dismantling process. This prioritized node is then removed from the

 doi:10.20944/preprints202505.2276.v2

network, and the procedure is iteratively repeated until the network is dismantled. See Figure 1 for a full breakdown of the LGD-NA framework.

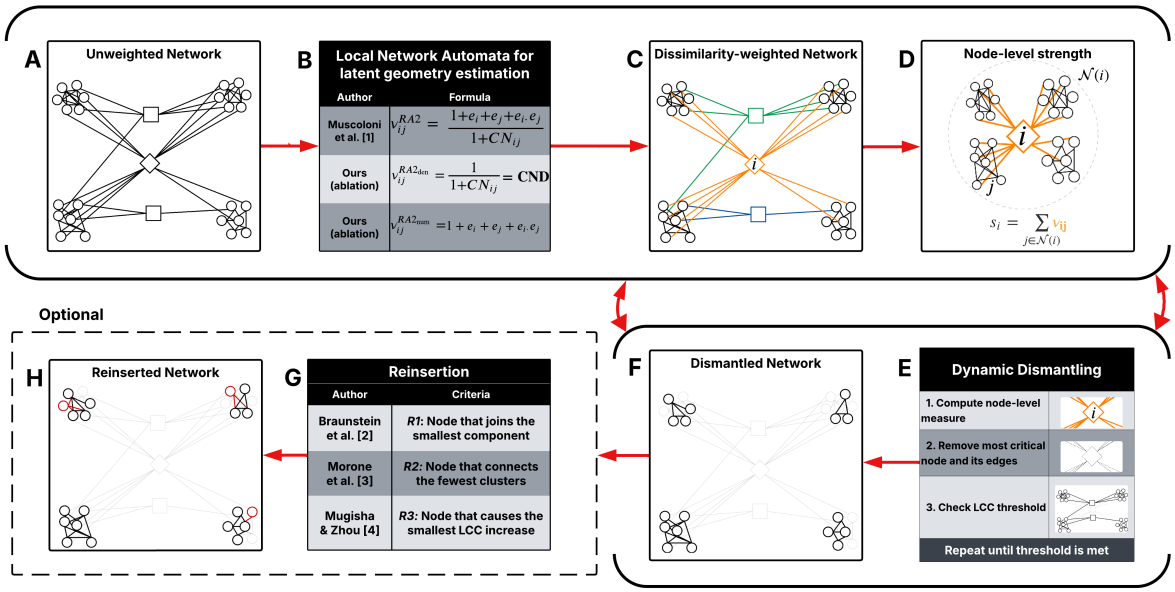

Figure 1. Overview of the LGD Network Automata framework. **A**: Begin with an unweighted and undirected network. **B**: Estimate latent geometry by assigning a weight ν_{ij} to each edge between nodes i and j using local latent geometry estimators (see Table A1). **C**: Construct a dissimilarity-weighted network based on these weights. **D**: Compute node strength as the sum of geometric weights to all neighbors in $\mathcal{N}(i)$:$s_i = \sum_{j\in\mathcal{N}(i)} \nu_{ij}$ **E–F**: Perform dynamic dismantling by iteratively computing node strengths, removing the node with the highest s_i and its edges, and checking whether the normalized size of the largest connected component (LCC) has dropped below a threshold. **G–H** (optional): Reinsert dismantled nodes using a selected reinsertion method (see Table A4).

3.1. Latent Geometry-Driven dismantling

Our first contribution is Latent Geometry-Driven (LGD) dismantling, where any function can be used to estimate edge weights that represent effective distances between nodes, capturing the network's underlying latent geometry. These inferred weights are used to construct a dissimilarity-weighted network, encoding a hidden geometric structure beneath the observable topology and allowing the dismantling process to prioritize nodes according to their geometric centrality in the latent manifold.

Latent geometric structures have been shown not only to explain key properties of complex networks, but also to support the understanding of dynamical processes such as navigation, routing, and epidemic spreading. Building on the idea that network geometry captures essential structural and dynamical properties of complex systems, LGD dismantling is guided by a geometric intuition about how nodes connect distant regions in the latent space. If two nodes are connected to many different nodes but have little overlap in their neighborhoods, they are likely to be far apart in the network's latent space. An edge between them, therefore, connects distant regions of the network. A node that has many such edges is central to holding the network together, as it links otherwise separate areas. We propose that removing those geometrically central nodes is an effective way to fragment the network.

Muscoloni et al. [1] also offered evidence that betweenness centrality can be used as a latent geometry estimator, hence, NBC is a global topology centrality measure which can be used for latent geometry-driven dismantling.

3.2. LGD Network Automata

Our second contribution is the introduction of a network automaton framework for LGD dismantling. In this framework, node importance is estimated by aggregating edge geometric weights into node strengths, and the network is dismantled iteratively by removing the nodes with the highest

 doi:10.20944/preprints202505.2276.v2

strength and all their edges. The underlying intuition is that nodes that connect to many external, non-overlapping regions are geometrically central and thus more structurally important, leading to higher strength values.

Formally, we begin with an undirected, unweighted network without isolated components. A network automaton rule, such as RA2, that is able to adopt local topology to estimate latent geometry, is applied to assign a weight ν_{ij} to the edge between node i and node j, representing the estimated geometric distance between the two nodes. We get a dissimilarity-weighted network from these edge weights. The strength s_i of node i is then calculated by summing the geometric weights of all its edges, that is, the weights to all its neighbors in the set $\mathcal{N}(i)$:

$$s_i = \sum_{j \in \mathcal{N}(i)} \nu_{ij}.$$

In this paper, we adopt three types of LGD network automata rules. The first rule is RA2, which was proposed by Muscoloni et al. [1] for hyperbolic network embedding purposes. The second rule is proposed in this study as an ablation test of the RA2 rule. It is the denominator of the RA2, which we call common neighbors dissimilarity (CND), defined as:

$$\nu_{ij} \to \mathrm{CND}(i,j) = \frac{1}{1 + \mathrm{CN}_{ij}}.$$

where CN_{ij} is the number of common neighbors between nodes i and j. Here, the lower the number of common neighbors two interacting nodes have, the more geometrically distant they are, and thus a higher edge weight is assigned between these two nodes.

The rationale for proposing a network automaton rule based only on the common neighbors denominator term of RA2 is to account for the mere attraction between a node and its neighbors. Neglecting the repulsion part associated with the external links (the numerator of RA2) makes sense in a dismantling task because any time we compute the common neighbors of a seed node with one of its neighbors, we indirectly account for the exclusion of nodes that are not in the topological neighborhood of the seed node. For completeness, we also investigate a third rule as an ablation test of RA2 in which we consider only the external links term in the RA2 numerator, expecting that the mere RA2 numerator should also work, but not as well as the common neighbor-based denominator. Indeed, a previous study offers evidence that common neighbors are among the topological features most associated with community organization and mesoscale network geometry [43].

4. Experiments

4.1. Evaluation Procedure

We evaluate all dismantling methods using a widely accepted procedure in the field of network dismantling [6]. For each method, nodes are removed sequentially according to the order it defines. After each removal, we track the normalized size of the Largest Connected Component (LCC), defined as the ratio $\mathrm{LCC}(x)/|N|$, where $|N|$ is the total number of nodes in the original network and $\mathrm{LCC}(x)$ is the number of nodes in the largest component after x removals. This process continues until the LCC falls below a predefined threshold. A commonly accepted threshold in dismantling studies is 10% of the original network size. To quantify dismantling effectiveness, we compute the Area Under the Curve (AUC) of the LCC trajectory throughout the removal process, which records the normalized LCC size at each step. A lower AUC indicates a more efficient dismantling, as it reflects an earlier and sharper disruption of network connectivity. The AUC is computed using Simpson's rule. See Figures A2, A4, A7, and A11 for visual illustrations of the LCC curve.

4.2. Optional Reinsertion Step

After reaching the dismantling threshold, we optionally perform a reinsertion step to reduce the dismantling cost, defined as the number of removals. Nodes are sequentially reinserted back

 doi:10.20944/preprints202505.2276.v2

into the network, one at a time, until the LCC of the remaining network just meets or exceeds the predetermined dismantling threshold. Reinsertion can significantly improve dismantling performance; recent work shows that simple heuristics with reinsertion can match or outperform complex algorithms that include reinsertion by default [44]. As a result, we enforce two constraints to ensure the reinsertion step does not override the original dismantling method: (1) reinsertion cannot reinsert all nodes to recompute a new dismantling order, and (2) reinsertion must use the reverse dismantling order as a tiebreak. If a method includes reinsertion by default, we also evaluate its performance without reinsertion for a fair comparison. See Table A4 for more information.

4.3. ATLAS Dataset

Our fourth contribution is the breadth and diversity of real-world networks tested in our experiments, demonstrating the generality and robustness of LGD-NA across domains and scales. We build an ATLAS of 1,475 real-world networks across 32 complex systems domains, which is the largest and most diverse collection of real-world networks to date used for testing in network dismantling studies. We first test all methods across networks of up to 5,000 nodes and 205,000 edges without reinsertion ($n = 1{,}296$), and 38,000 edges with reinsertion ($n = 1{,}237$). To assess the practical running time of the best performing methods, we evaluate NBC and RA2 on even larger networks of up to 23,000 nodes and 507,000 edges ($n = 1{,}475$).

Current state-of-the-art dismantling algorithms have been evaluated on no more than 57 real-world networks (see Table 1), with most algorithms tested on fewer than a dozen. Our experiments cover 1,475 networks, representing a substantial expansion.

A key aspect of our ATLAS dataset is the diversity of network types (see Table 2). We test across 32 different complex systems domains, ranging from protein-protein interaction (PPI) to power grids, international trade, terrorist activity, ecological food webs, internet systems, brain connectomes, and road maps. Since fields vary in both the number of networks and their characteristics, we evaluate dismantling methods using a mean field approach, ensuring that fields with more networks do not dominate the overall evaluation. Also, because dismantling performance varies in scale across fields, we compute a mean field ranking to make results comparable across domains. Specifically, we rank all methods within each field based on their average AUC, then compute the mean of these rankings across all fields.

Table 2. Summary of real-world networks tested in this paper, see Table A5 for more information.

Field	Subfields	Types	Networks
Biomolecular	5	PPI, Genetic, Metabolic, Molecular, Transcription	27
Brain	1	Connectome	529
Covert	2	Covert, Terrorist	89
Foodweb	1	Foodweb	71
Infrastructure	7	Flight, Nautical, Power grid, Rail, Road, Subway, Trade	314
Internet	1	Internet	206
Misc	8	Citation, Copurchasing, Game, Hiring, Lexical, Phone call, Software, Vote	38
Social	7	Coauthorship, Collaboration, Contact, Email, Friendship, Social network, Trust	201
Total	**32**		**1,475**

 doi:10.20944/preprints202505.2276.v2

4.4. LGD-NA Performance and Comparison to Other Methods

We compare our Latent Geometry-Driven Network Automata (LGD-NA) framework against the best-performing dismantling algorithms in the literature. Main results are shown in Figure 2.

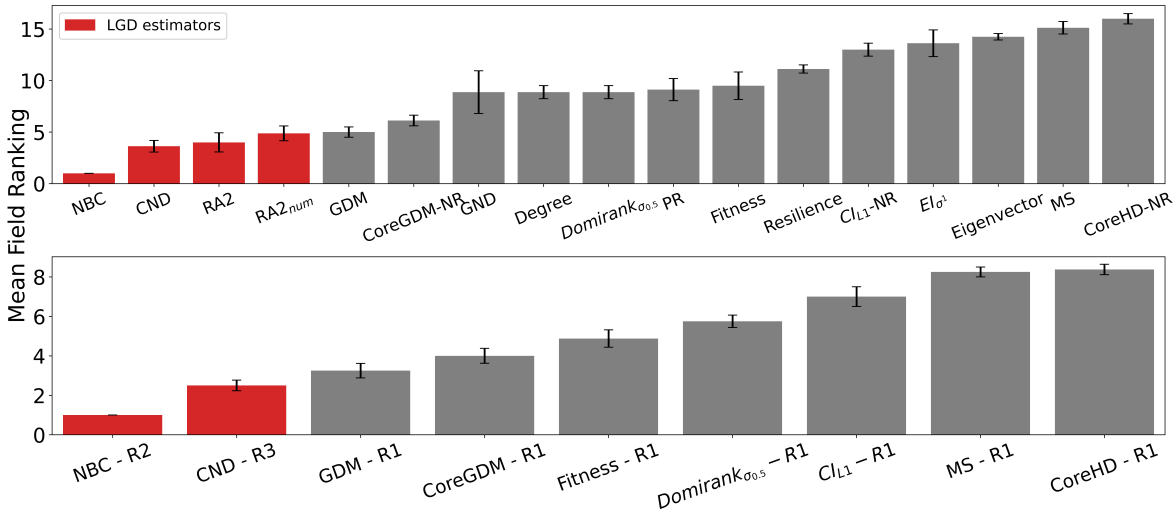

Figure 2. Mean field ranking for each dismantling method without reinsertion (n = 1,296; upper panel) and with reinsertion (n = 1,237; lower panel), for dynamic dismantling. All methods are described in the Appendix 5 and in Tables A1, A2, and A3. In the lower panel, a subset of the best-performing methods from each category is paired with their respective best-performing reinsertion strategy (see Table A4). Methods based on latent geometry are shown in red. For EI, CI, and Domirank, we report results with their optimal parameter configurations. NR denotes variants where the original reinsertion step was disabled. Error bars indicate the standard error of the mean (SEM).

First, we find that all latent geometry network automata, NBC, RA2, and its variants, achieve top dismantling performance, both with and without reinsertion. These findings show that estimating the latent geometry of a network effectively reveals critical nodes for dismantling, confirming our first contribution.

For each method, we evaluate three reinsertion strategies and report the best result. We show in Figure A3 that using different reinsertion methods does not change the mean field ranking of the dismantling methods, and in Figure A5 that the improvement in performance varies across fields and reinsertion methods (see Figure A4 for an illustrative example). We also adopt a dynamic dismantling process for the network automata rules and all centrality measures, where we recompute the scores after each dismantling step, as it consistently outperforms the static variant (see Figure A6 for an example of the improvements for CND and Figure A7 for an illustrative example).

Second, we find that local network automata rules RA2, CND, and RA2_{num}, which adopt only the local network topology around a node, are highly effective. In particular, RA2 and its variants consistently outperform all other non-latent geometry-driven dismantling algorithms, including those relying on global topological measures or machine learning. This confirms our second contribution. See Figure A2 for illustrative examples where the local network automata rules outperform NBC. In addition, Figure A9 confirms that these network automata rules effectively approximate latent geometric distances in hyperbolic space, as the more radially central nodes in the hyperbolic nPSO networks are indeed those with the highest RA2 measures.

Third, we find that the simplest RA2 variant, based solely on inverse common neighbors, which we refer to as common neighbor dissimilarity (CND), achieves the best performance among all local network automata rules. This is our third contribution and demonstrates that even minimal local topology-based information can effectively approximate latent geometry useful for effective dismantling.

 doi:10.20944/preprints202505.2276.v2

LGD-NA outperforms all other methods, is robust to the inclusion or omission of reinsertion, and is validated across a large and diverse set of networks. These results strongly demonstrate the practical reliability of our latent geometry-driven dismantling framework, LGD-NA.

4.5. GPU Acceleration of LGD-NA for Large-Scale Dismantling

We implement GPU acceleration for all three LGD-NA variants by reformulating the required computations as matrix operations. On large networks, this enables a significant speedup in running time. When comparing RA2 and NBC, on the largest network, GPU-accelerated RA2 is 130 times faster than its CPU counterpart, highlighting the inefficiency of matrix multiplication on CPU. It is also over 63 times faster than NBC running on CPU, thanks to our GPU-optimized implementation. For NBC, we report only CPU running time, as its GPU implementation did not yield any speedup, as explained in Appendix H. Note that NBC on CPU remains faster than RA2 on CPU, again due to the limitations of CPU-based matrix operations.

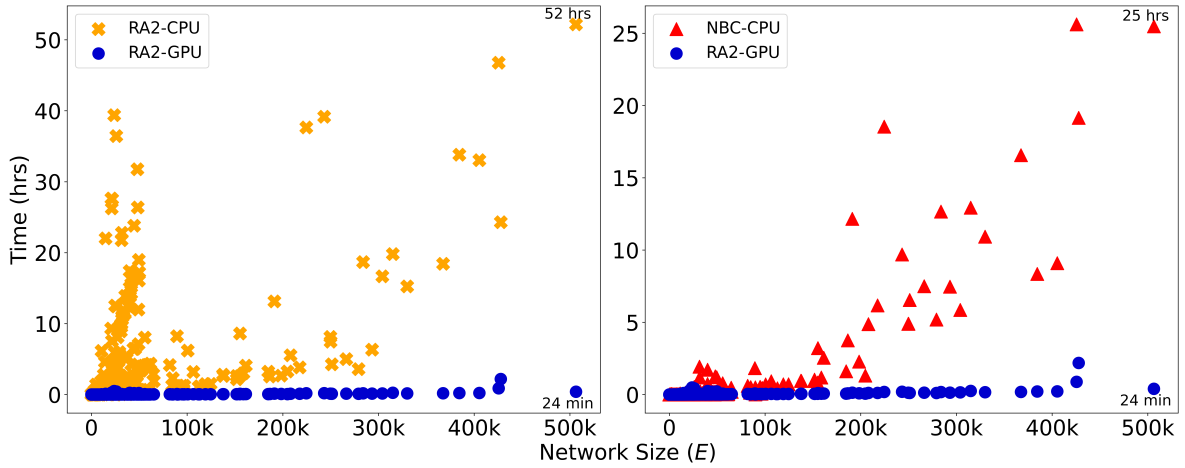

Figure 3. Runtime (in hours) is plotted against network size, measured by the number of edges, E, for dynamic dismantling. The annotated time indicates the runtime for the largest network. Evaluated on networks of up to 23,000 nodes and 507,000 edges ($n = 1{,}475$). See Figure A10 for CND.

Overall, while NBC achieves better dismantling performance, its high computational cost makes it impractical for large-scale use. In contrast, thanks to our GPU-optimized implementation, our local latent geometry estimators based on network automata rules are the only viable option for efficient dismantling at scale.

Here, we look at the details of our matrix operations for the LGD-NA measures. First, the common neighbors matrix is computed as

$$\mathbf{CN}_{\mathrm{L2}} = \mathbf{A} \circ (\mathbf{A}^2)$$

where $\mathbf{A}$ is the adjacency matrix and $\circ$ denotes element-wise multiplication. Here, $\mathbf{A}^2$ counts the number of paths of length two (i.e., common neighbors) between all node pairs. The Hadamard product with $\mathbf{A}$ ensures that values are only retained for existing edges.

Next, we compute the number of external links a node has relative to each of its neighbors. Given the degree matrix $\mathbf{D}$, the external degree matrix is:

$$\mathbf{E}_{\mathrm{L2}} = \mathbf{A} \circ (\mathbf{D} - \mathbf{CN}_{\mathrm{L2}} - \mathbf{A})$$

Each entry (i, j) of $\mathbf{E}_{\mathrm{L2}}$ represents the external degree of node i with respect to node j: the number of neighbors of i that are neither connected to j nor directly connected to j itself. Non-edges are zeroed out. These matrices allow efficient construction of RA2 and its variants using only matrix operations.

The time complexity is $\mathcal{O}(N^3)$, with the common neighbor matrix being the dominant operation, for dense graphs, and $\mathcal{O}(Nm)$ for sparse graphs, N being the number of nodes and m the number of

 doi:10.20944/preprints202505.2276.v2

links. On CPU, matrix multiplication is typically memory-bound and limited by sequential operations. GPUs, however, are optimized for matrix operations, leveraging thousands of parallel threads. This results in a substantial speedup when implementing the GPU version.

4.6. Leveraging CND Explainability to Engineer Network Robustness

A key advantage of our LGD-NA framework is its explainability. Indeed, we can directly explain why any of our network-automata-based and latent-geometry-driven measures prioritize specific nodes for dismantling. CND, our most performant network automata rule for dismantling, makes this explainability even more straightforward and shows that the vulnerability of a node is strongly related to the number of links its neighbors share with one another. The higher this number, the more common neighbors exist between the adjacent nodes of a vulnerable target node. This means that, to enhance the robustness of the network to the failure of a critical node, we should simply increase the number of links between its adjacent nodes.

The strategy is as follows. First, identify the nodes with the highest dismantling scores according to a given measure. Here, we consider NBC, a global shortest-path count-based measure, and CND, a local topology common-neighbor-based network automata measure, because they use different rationales to estimate critical nodes and are the two best-performing measures in this study. Second, for these critical nodes, add new links between their adjacent nodes that are not already connected to each other.

We validate this reinforcement strategy in Table A7 and Figure A8. We clearly show that adding links between the adjacent nodes of the most critical nodes significantly increases the AUC—and therefore the robustness—by 36% to 95% for 1% of added links, and by 59% to 259% for 10% of added links. Remarkably, by reinforcing only the top 1% of nodes, we increase network robustness regardless of the dismantling method used—whether it is our CND or NBC.

5. Conclusion

We introduced Latent Geometry-Driven Network Automata (LGD-NA), a framework that achieves state-of-the-art network dismantling using only local topological information. By applying simple network automata rules to estimate a network's latent geometry, LGD-NA identifies critical nodes with significant speed advantages over global methods like Node Betweenness Centrality (NBC). Across 1,475 real-world networks and 32 complex systems domains, it consistently outperforms all existing methods, including those based on machine learning (e.g., Graph Neural Networks). Notably, our minimalistic Common Neighbor Dissimilarity (CND) measure matches NBC's efficacy while being orders of magnitude faster. While Muscoloni et al. [1] established RA2 as a strong local estimator for tasks like embedding and community detection, we show that for dismantling, even simpler network automata rules like CND are applicable and effective. Leveraging the explainability of CND, we also introduce a novel strategy to engineer network robustness by adding targeted links, significantly enhancing robustness. This work establishes latent geometry as a powerful and efficient principle for both explaining vulnerabilities and engineering stronger networks.

Supplementary Materials: The following supporting information can be downloaded at the website of this paper posted on Preprints.org

Acknowledgments: This work was supported by the Zhou Yahui Chair Professorship award of Tsinghua University (to CVC), the National High-Level Talent Program of the Ministry of Science and Technology of China (grant number 20241710001, to CVC).

 doi:10.20944/preprints202505.2276.v2

Appendix A. Latent Geometry Estimators

Table A1. Comparison of latent geometry estimators and their variants. ν_{ij} is the weight of the link between nodes i and j; e_i and e_j denote the number of external links of nodes i and j, respectively; CN_{ij} is the number of common neighbors shared by i and j. *Information Locality* denotes the type of structural information required to assign a score to each node for dismantling. *Time Complexity* denotes the time complexity for dynamic dismantling using each estimator on sparse graphs, without reinsertion. N: number of nodes. m: number of links.

Estimator	Author	Year	Formula	Information Locality	Time Complexity
Repulsion Attraction 2	Muscoloni et al. [1]	2017	$\nu_{ij}^{RA2} = \frac{1+e_i+e_j+e_i.e_j}{1+CN_{ij}}$	Local	$\mathcal{O}(N(Nm))$
RA2 denominator-ablation (CND)	Ours	2025	$\nu_{ij}^{RA2_{\mathrm{den}}} = \frac{1}{1+CN_{ij}} = CND$	Local	$\mathcal{O}(N(Nm))$
RA2 numerator-ablation	Ours	2025	$\nu_{ij}^{RA2_{\mathrm{num}}} = 1 + e_i + e_j + e_i.e_j$	Local	$\mathcal{O}(N(Nm))$

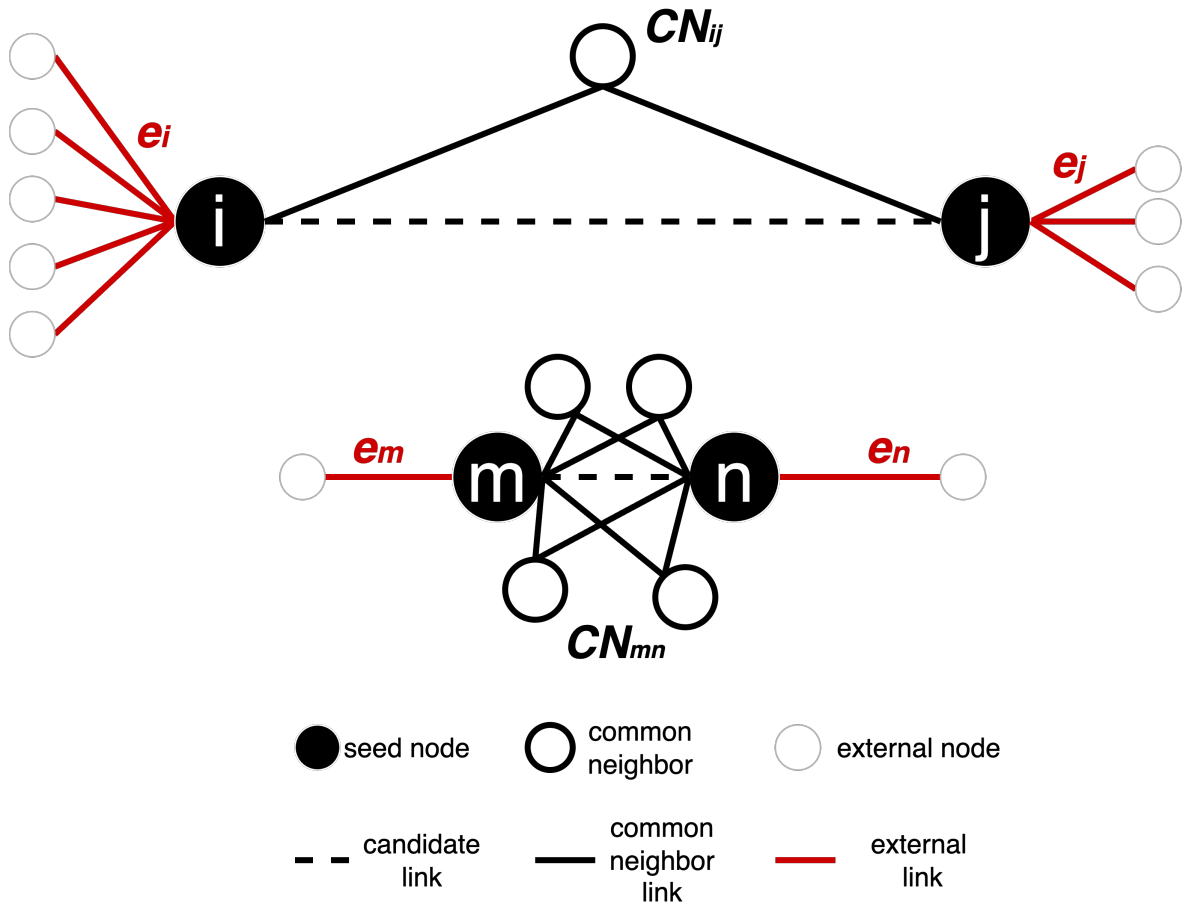

Figure A1. Illustration of how RA2 measures are computed on two toy networks. Seed nodes are shown in black; common neighbors (CN) are shown in white with a black border, and external nodes are white with a grey border. The dashed line is the edge that is being assigned a weight. External links e denote the number of edges connecting a node to nodes outside its CN set, here in red. In black, the links to common neighbors. For the link ν_{ij} in the top network, $e_i = 5$, $e_j = 3$, and $\mathrm{CN_{ij}} = 1$. For the link ν_{mn} in the bottom network, $e_m = 1$, $e_n = 1$, and $\mathrm{CN_{mn}} = 4$.

Latent Geometry-Driven Network Automata rule.

Figure A1 illustrates how RA2-based network automata rules assign edge weights by estimating geometric distances using only local topological features. The two toy subnetworks demonstrate how the RA2 rule and its variants distinguish between geometrically distant and close node pairs. In the top subnetwork, nodes i and j have only one common neighbor and are each connected to many external

 doi:10.20944/preprints202505.2276.v2

nodes ($e_i = 5, e_j = 3$), indicating a weak integration in a local community and stronger connectivity to distinct parts of the network. According to the Repulsion-Attraction rule, this suggests a larger latent distance due to high repulsion and low attraction. In contrast, in the bottom subnetwork, nodes m and n share four common neighbors and have only one external link each ($e_m = 1, e_n = 1$). This pattern indicates a stronger local community and a higher likelihood that the nodes are geometrically close in the latent space, with a lower dissimilarity score. These examples highlight how latent geometry-driven RA2-based network automata rules estimate hidden distances: fewer common neighbors and more external links suggest geometrical separation, while many common neighbors and few external links imply proximity in the latent manifold.

Why is RA2 a latent geometry estimator?

In geometric networks, nearby nodes form dense, closed neighborhoods: they share many common neighbors (high CN_{ij}) and have few "external" links (small e). Distant node pairs show the opposite pattern. The Repulsion-Attraction rule 2 (RA2) captures these patterns in their formulation: any RA variant that decreases with CN_{ij} and increases with external connectivity, e is therefore monotonic with latent distance: small values indicate proximity; large values indicate separation. Crucially, this relies on topological proximity, not a specific geometric space, so it applies across hyperbolic, Euclidean, or elliptic latent geometries. The only assumption to apply this estimator of underlying geometry is that the topology displays:

- node heterogeneity (meaning that the node degree distribution displays a standard deviation different from zero).
- homophily (similar nodes link together), for instance geometric proximity in latent space causes nodes that are geometrically close have overlapping neighborhoods.

We can aggregate RA2 to score node criticality by summing its pairwise RA2 to neighbors. This turns local edge-level "distance" into a node-level bridging load:

- Few adjacent nodes (neighbors) with mostly short links yields a small RA2. This node is peripheral and non-critical; removal has little global effect.
- Many neighbors with mostly short links still yields a modest RA2 (short links contribute little). The node is locally redundant; removal is buffered by community structure.
- Many neighbors with a mix of short and long links yields a large RA2 because long links carry high RA2. The node simultaneously anchors a local community and bridges distant regions; removing it is likely to disconnect communities and degrade global connectivity.
- Few neighbors with many long links (rare under geometric attachment) still yields a large RA2; such nodes are likewise critical inter-community hubs.

As a result, RA2 encodes latent separation from purely local topology. Summing RA2 over a node's incident edges ranks nodes by how much long-range connectivity they support. Dismantling the highest-scoring nodes precisely targets those bridges whose removal most effectively fragments the network.

Time Complexity.

We analyze the time complexity for the full dynamic dismantling process (excluding reinsertion) for the latent geometry-driven network automata rules in Table A1, where dynamic means recomputing the dismantling measure after each node removal. For RA2 and its variants, the dominant operation is the computation of the common neighbor (CN) matrix. This operation has a time complexity of $\mathcal{O}(N^3)$ for dense graphs and $\mathcal{O}(Nm)$ for sparse graphs, where N is the number of nodes and m is the number of links. Assuming N dismantling steps in the worst-case scenario, the overall time complexity becomes $\mathcal{O}(N(Nm))$ for sparse graphs. The assumption of N dismantling steps applies to all the time complexity analyses of dynamic dismantling methods.

 doi:10.20944/preprints202505.2276.v2

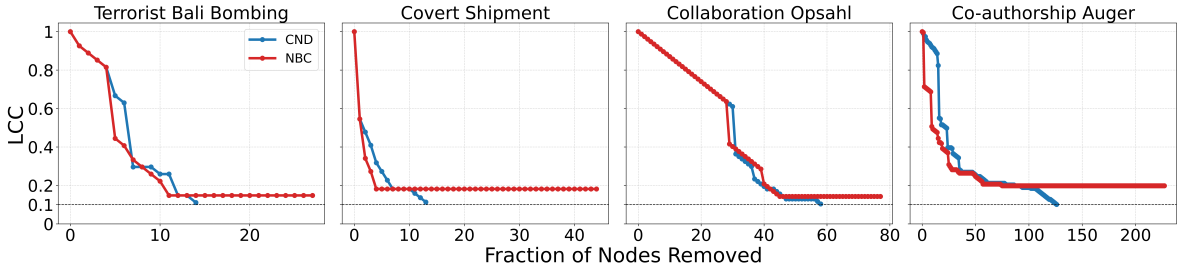

Figure A2. Dynamic dismantling process on example networks comparing local network automata rule RA2 and its variants versus NBC, when the former outperforms NBC in terms of AUC. The plot shows the normalized size of the largest connected component (LCC) as a function of the fraction of nodes removed, with a target LCC threshold of 10%. The final evaluation metric is the Area Under the Curve (AUC) of the LCC trajectory.

Appendix B. Topological Centrality Measures

Table A2. Comparison of topological centrality measures and the associated time complexity for dynamic dismantling using each centrality measure. *Information Locality* denotes the type of structural information required to assign a score to each node. *Time Complexity* denotes the time complexity for dynamic dismantling using each centrality measure on sparse graphs, without reinsertion. N: number of nodes. m: number of links.

Measure	Author	Year	Type	Information Locality	Time Complexity
Degree			Degree-based	Local	$\mathcal{O}(N \log N)$
Eigenvector	Bonacich [40]	1972	Walks-based	Global	$\mathcal{O}(N(N+m))$
Node Betweenness (NBC)	Freeman [21]	1977	Shortest path-based	Global	$\mathcal{O}(N(Nm))$
PageRank (PR)	Page et al. [41]	1999	Random walk-based	Global	$\mathcal{O}(N(N+m))$
Resilience	Zhang et al. [33]	2020	Resilience-based	Global	$\mathcal{O}(N(N+m)$
Domirank	Engsig et al. [22]	2024	Fitness-based	Global	$\mathcal{O}(N(N+m)$
Fitness	Servedio et al. [23]	2025	Fitness-based	Global	$\mathcal{O}(N(N+m)$

Time Complexity.

We analyze the time complexity of dynamic dismantling (excluding reinsertion) for the topological centrality measures used in our experiments, summarized in Table A2. As before, the analysis assumes N dismantling steps in the worst-case scenario. For degree, the score update after each removal is local and can be done in $\mathcal{O}(\log N)$ time using a binary heap. For NBC, we use Brandes' algorithm [45], which computes betweenness centrality in $\mathcal{O}(Nm)$ time per step for unweighted networks. Eigenvector, PageRank, Resilience, Domirank, and Fitness all rely on matrix-vector multiplications, which has a time complexity of $\mathcal{O}(m)$. We also add the term N, which represents the overhead of looping over nodes to update or normalize the resulting vector at each iteration. This leads to a total per-step cost of $\mathcal{O}(N+m)$. We also omit the constant k for Eigenvector, PageRank, and Fitness centrality, which represents the number of iterations these methods perform. In practice, reaching full convergence to a single optimal solution is often computationally infeasible; this is why a fixed number of k iterations is typically defined.

 doi:10.20944/preprints202505.2276.v2

Appendix C. Statistical and Machine Learning Network Dismantling

Table A3. Comparison of dismantling algorithms [6]. *Information Locality* denotes the type of structural information required to assign a score to each node. *Dynamicity* indicates whether scores are recomputed after each removal. *Reinsertion* specifies whether the algorithm includes a reinsertion step after dismantling. *Time Complexity* denotes the time complexity of the method on sparse graphs, without reinsertion. N: number of nodes. m: number of links. h: number of attention heads. T: maximal diameter of the trees in the forest for BPD and MS. ϵ is a small constant used in spectral partitioning operations. *Included* states whether the method was run in our experiments; if not, a brief reason is provided.

Algorithm	Type	Author	Year	Information Locality	Dynamicity	Reinsertion	Time Complexity	Included
Collective Influence (CI)	Influence maximization	Morone et al. [3]	2016	Local	Dynamic	Yes	$\mathcal{O}(N \log N)$	Yes
Belief propagation-guided decimation (BPD)	Message passing-based decycling	Mugisha and Zhou [4]	2016	Global	Dynamic	Optional	$\mathcal{O}(mT)$	No - Code missing
Min-Sum (MS)	Message passing-based decycling	Braunstein et al. [2]	2016	Global	Dynamic	Yes	$\mathcal{O}(mT) + \mathcal{O}(N(\log N + T))$	Yes
Generalized Network Dismantling (GND)	Spectral partitioning	Ren et al. [32]	2019	Global	Dynamic	Optional	$\mathcal{O}(N \log^{2+\epsilon} N)$	Yes
CoreHD	Degree-based decycling	Zdeborová et al. [30]	2016	Global	Dynamic	Yes	$\mathcal{O}(N)$	Yes
Explosive Immunization (EI)	Explosive percolation	Clusella et al. [31]	2016	Global	Dynamic	No	$\mathcal{O}(N \log N)$	Yes
FINDER	Machine learning	Fan et al. [42]	2020	Global	Dynamic	Optional	$\mathcal{O}(N(1 + \log N) + m)$	No - Code outdated
Graph Dismantling Machine (GDM)	Machine learning	Grassia et al. [34]	2021	Global	Static	Optional	$\mathcal{O}(h(N + m))$	Yes
CoreGDM	Machine learning	Grassia and Mangioni [35]	2023	Global	Static	Yes	$\mathcal{O}(h(N + m))$	Yes

Table A3 is adapted and extended from Table 1 of Artime et al. [6], a recent and comprehensive review which has become a key reference in the field of network dismantling. The majority of these algorithms were included in our experiments, with the exception of BPD and FINDER due to unavailable or outdated code, respectively.

 doi:10.20944/preprints202505.2276.v2

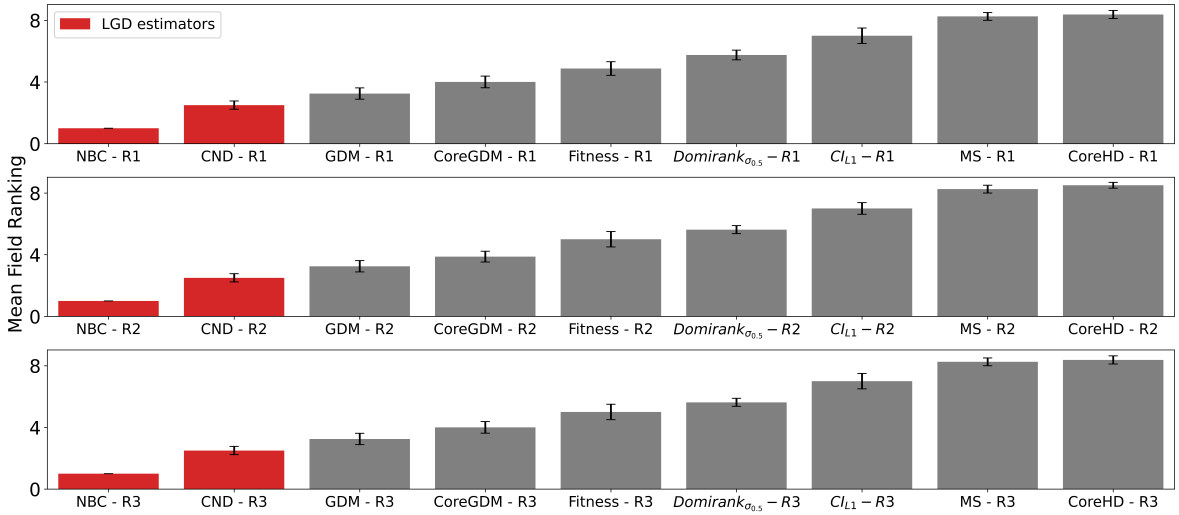

Figure A3. Mean field ranking for a subset of the best-performing methods from each category with each reinsertion method (R1, R2, R3) ($n = 1{,}237$). Methods based on latent geometry are shown in red. All LGD and topological centrality measures use dynamic dismantling. Error bars indicate the standard error of the mean (SEM). Method acronyms are defined in Tables A1, A2, and A3.

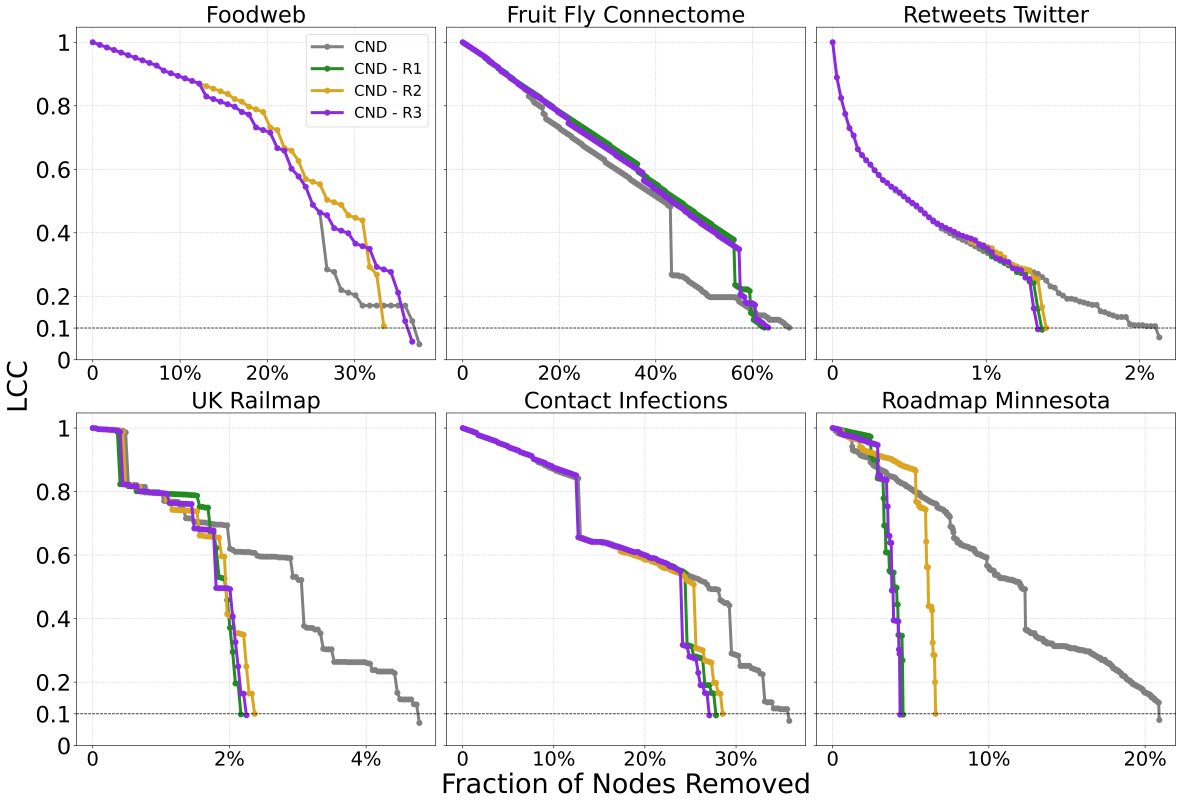

Figure A4. Dynamic dismantling process on example networks comparing CND with and without reinsertion. The plot shows the normalized size of the largest connected component (LCC) as a function of the fraction of nodes removed, with a target LCC threshold of 10%. The final evaluation metric is the Area Under the Curve (AUC) of the LCC trajectory.

Appendix D. Reinsertion Methods

Reinsertion was originally introduced in the context of immunization as a reverse process: starting from a fully dismantled network, nodes are reinserted one by one, each time selecting the node whose addition causes the smallest increase in the largest connected component (LCC) [46]. This reversed sequence then defines an effective dismantling order. In subsequent studies, reinsertion has been used

 doi:10.20944/preprints202505.2276.v2

as a post-processing step to improve dismantling outcomes [6]: the network is first dismantled by a given method, and nodes are reinserted until the LCC reaches the dismantling threshold. This reduces the dismantling cost while preserving the original attack target.

In this work, dismantling cost is defined as the number of nodes removed from the network. The reinsertion step aims to directly minimize this cost by reintroducing nodes that were initially removed but found to be unnecessary for achieving the dismantling objective. It's important to note that while reinsertion reduces the number of physical removals, it does introduce a higher computational cost as it's a post-processing step performed after the initial dismantling. However, the primary objective is to minimize this physical intervention, as in many real-world scenarios, the logistical and financial implications of physically removing network components (e.g., infrastructure) far outweigh the computational resources expended during the optimization phase. This is why we compare all methods with and without the reinsertion step.

Several reinsertion criteria have been proposed: Braunstein et al. [2] select the node that ends up in the smallest resulting component after reinsertion; Morone et al. [3] choose the node that reconnects the fewest components; Mugisha and Zhou [4] select the node that causes the smallest LCC increase. See Table A4 for a full comparison.

Reinsertion can greatly enhance dismantling performance. However, recent work shows that this step can overpower the dismantling algorithm itself, allowing weak methods to appear effective when paired with reinsertion [44]. To address this, we enforce two constraints to ensure fair comparisons and prevent reinsertion from dominating the dismantling process:

1. Reinsertion must stop once the LCC exceeds the dismantling threshold. Recomputing a new dismantling order by reinserting all nodes is not allowed.
2. Ties in the reinsertion criterion must be broken by reversing the dismantling order: nodes removed later are prioritized.

These rules ensure that reinsertion complements rather than overrides the dismantling process, preserving the integrity of the original method.

In our experiments, we implement three reinsertion methods, adapted from prior work, here we explain which part of their method we change for our experiments. Those changes are marked with an asterisk (*) in Table A4.:

- **R1** [2]: We replace their original tiebreak (smallest node index) with reverse dismantling order.
- **R2** [3]: We apply the LCC stopping condition. Originally, all nodes are reinserted to compute a new dismantling sequence.
- **R3** [4]: We apply reverse dismantling order as the tiebreak, as no rule is defined in their paper, and their code is unavailable.

R3 is the most similar to the reverse immunization method proposed by Schneider et al. [46], where nodes are added back one by one based on minimal LCC growth. In their original method, ties are broken by selecting the node with the fewest connections to already reinserted nodes; if multiple candidates remain, one is chosen at random.

We note that reinsertion typically reduces the number of removals but does not always lead to a lower AUC. Since the trajectory of the LCC changes with reinsertion, the dismantling process may reach the threshold faster, improving AUC. However, this is not guaranteed, as we see in the first two subplots of Figure A4 for the Foodweb and Fruit Fly Connectome networks. The methods with reinsertion arrive at the dismantling threshold in fewer number of removals, but the change in the LCC curve results in a worse final AUC.

We also see that the reduction in AUC is not proportional to the reduction in the number of removals, as seen in Figure A5 for CND. Indeed, reinsertion, by definition, reinserts nodes that were ultimately unnecessary for the dismantling process to reach its target.

 doi:10.20944/preprints202505.2276.v2

Time Complexity.

We report the total time complexity of each reinsertion method over the full reinsertion process in Table A4, assuming all dismantled nodes are considered for reinsertion for every step and that all nodes are reinserted. Candidates for reinsertion are denoted as r. As a result, we multiply the per-step cost of updating for each method by the total number of reinsertion candidates. $k_{\max}$ is the maximum number of components a node can connect to, equal to the maximum degree in the original graph, and C' is the maximum size of any connected component during the reinsertion phase. For R1, the candidate node that ends up in the smallest resulting component is selected. Reinserting a node may merge up to $k_{\max}$ components, each of size at most C', requiring an update of at most $k_{\max} \cdot C'$ nodes. These updates are tracked in a binary heap of size r, where at maximum $k_{\max} \cdot C'$ nodes have to be updated, giving a cost of $\log(k_{\max} \cdot C')$ per update. The per-step cost is therefore $\mathcal{O}(k_{\max} \cdot C' \log(k_{\max} \cdot C'))$. R2 selects the node that connects the fewest existing components. Unlike R1, it requires inspecting not only the components merged by the candidate node, but also the neighbors of the affected neighbors. This increases the complexity by a factor of $k_{\max}$, resulting in a per-step time complexity of $\mathcal{O}(k_{\max}^2 \cdot C' \log(k_{\max}^2 \cdot C'))$. R3 evaluates each candidate by explicitly computing the resulting LCC size after reinsertion. Each evaluation requires a graph traversal to recompute connected components, which takes $\mathcal{O}(N+m)$ time on sparse graphs. This has to be done for each reinsertion candidate, at every step, so $\mathcal{O}(r^2(N+m))$,

Table A4. Comparison of reinsertion methods. *Criteria* defines the criterion for selecting which node to reinsert. *Tiebreak* specifies how ties are resolved. *LCC Condition* indicates whether all dismantled nodes are reinserted or if reinsertion stops once the predefined LCC threshold is reached. *Time Complexity* denotes the time complexity of each reinsertion method on sparse graphs, for the whole reinsertion process. N: number of nodes. m: number of links. r: set of reinsertion candidates. $k_{\max}$: maximum degree in the original graph G. C': maximum size of any connected component during the reinsertion phase. *Used In* lists the methods that use each method, in bold, the dismantling method that originally proposed that reinsertion method. An asterisk (*) marks components of the reinsertion method that were modified in our study, as detailed in Appendix D.

Name	Author	Year	Criteria	Tiebreak	LCC Condition	Time Complexity	Used In
R1	Braunstein et al. [2]	2016	Node that ends up in the smallest component	Reverse dismantling order*	Yes	$\mathcal{O}(r(k_{\max} \cdot C' \cdot \log(k_{\max} \cdot C')))$	**MS**, CoreGDM, CoreHD, GDM, GND
R2	Morone et al. [3]	2016	Node that connects to the fewest clusters	Reverse dismantling order	Yes*	$\mathcal{O}(k_{\max}^2 \cdot C' \cdot \log(k_{\max}^2 \cdot C'))$	**CI**
R3	Mugisha and Zhou [4]	2016	Node that causes the smallest increase in LCC size	Reverse dismantling order*	Yes	$\mathcal{O}(r^2(N+m))$	**BPD**

 doi:10.20944/preprints202505.2276.v2

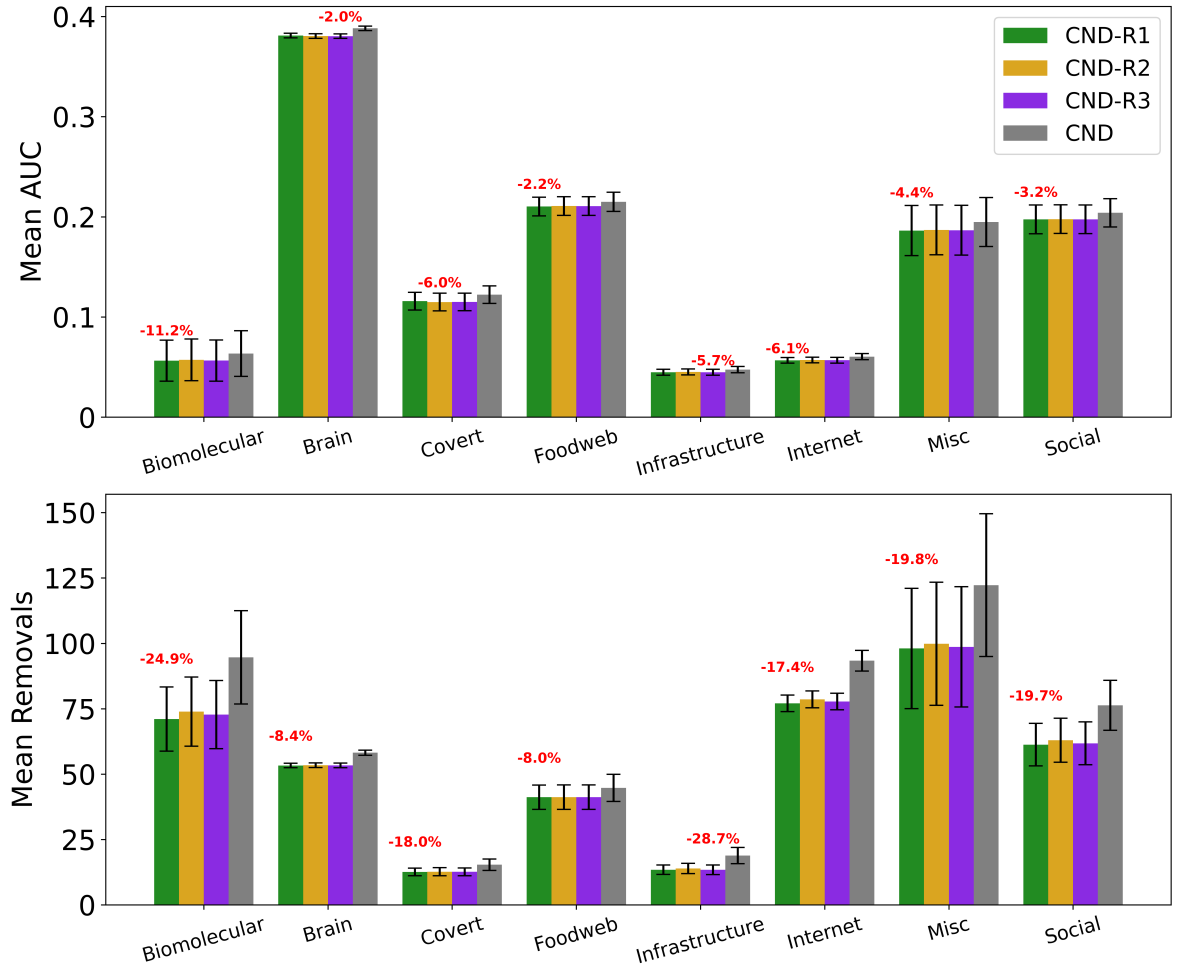

Figure A5. Mean AUC and number of removals by field for CND without reinsertion and with each reinsertion method (R1, R2, R3) ($n = 1{,}237$ for all methods). Error bars represent the standard error of the mean (SEM). Red text indicates the percentage improvement achieved by using the best-performing reinsertion method for each field.

Table A5. Full summary statistics of the ATLAS networks used in this study, averaged by field: number of subfields and networks, average number of nodes $\langle N \rangle$, number of edges $\langle E \rangle$, density $\langle \rho \rangle$, mean degree $\langle \langle d \rangle \rangle$, characteristic path length $\langle \ell \rangle$, assortativity $\langle r \rangle$ [47], transitivity $\langle T \rangle$, mean local clustering coefficient $\langle \langle \text{Loc. CC} \rangle \rangle$, maximum k-core $\langle k_{\max} \rangle$, average k-core $\langle \langle k \rangle \rangle$, LCP-corr $\langle LCP_{corr} \rangle$ [48], and modularity $\langle Q \rangle$ [49]

Field	Biomolecular	Brain	Covert	Foodweb	Infrastructure	Internet	Misc	Social	**Total**
Subfields	5	1	2	1	7	1	8	7	**32**
Networks	27	529	89	71	314	206	38	201	**1,475**
$\langle N \rangle$	2,997	97	107	117	664	5,708	2,880	3,267	
$\langle E \rangle$	11,855	1,535	266	1,087	1,332	19,601	19,921	53,977	
$\langle \rho \rangle$	0.01	0.34	0.17	0.16	0.07	0.01	0.07	0.11	
$\langle \langle d \rangle \rangle$	6.7	28.3	5.7	15.2	4.9	7.5	14.1	26.9	
$\langle \ell \rangle$	4.4	1.7	3	2.2	9.9	3.4	3.5	3.5	
$\langle r \rangle$	-0.21	-0.03	-0.15	-0.28	-0.52	-0.22	-0.07	-0.05	
$\langle T \rangle$	0.06	0.55	0.39	0.19	0.06	0.11	0.22	0.29	
$\langle \langle Loc.CC \rangle \rangle$	0.13	0.63	0.46	0.22	0.11	0.31	0.34	0.36	
$\langle k_{\max} \rangle$	10.6	20.1	5.9	12.8	4.9	25	21.6	25.7	
$\langle \langle k \rangle \rangle$	3.6	17.5	4.2	9.2	3	4	8.3	15.4	
$\langle LCP_{corr} \rangle$	0.66	0.97	0.76	0.67	0.15	0.94	0.85	0.77	
$\langle Q \rangle$	0.59	0.25	0.48	0.26	0.46	0.5	0.49	0.5	

 doi:10.20944/preprints202505.2276.v2

Table A6. Number and size of real-world networks tested by dismantling algorithms. N denotes the number of nodes, E the number of edges.

Algorithm	Year	Networks	N_{max}	E_{max}
Collective Influence (CI) [3]	2016	2	14M	51M
CoreHD [30]	2016	12	1.7M	11M
Explosive Immunization (EI) [31]	2016	5	50K	344K
Min-Sum (MS) [2]	2016	2	1.1M	2.9M
Generalized Network Dismantling (GND) [32]	2019	10	5K	17K
Resilience Centrality [33]	2020	4	1K	14K
Graph Dismantling Machine (GDM) [34]	2021	57	1.4M	2.8M
CoreGDM [35]	2023	15	79K	468K
Domirank Centrality [22]	2024	6	24M	58M
Fitness Centrality [23]	2025	5	297	4K
LGD-NA	**2025**	**1,475**	**23K**	**507K**

Appendix E. Dynamic & Static Dismantling

In static dismantling, node scores are computed once at the beginning and are then removed in descending order of importance until the dismantling threshold is reached. In contrast, dynamic dismantling recomputes the scores after each removal. As shown in Figure A6, with CND given as an example, dynamic dismantling consistently outperforms static dismantling across all fields. Dynamic variants achieve lower AUC and fewer removals in every case, confirming the advantage of score recomputation.

 doi:10.20944/preprints202505.2276.v2

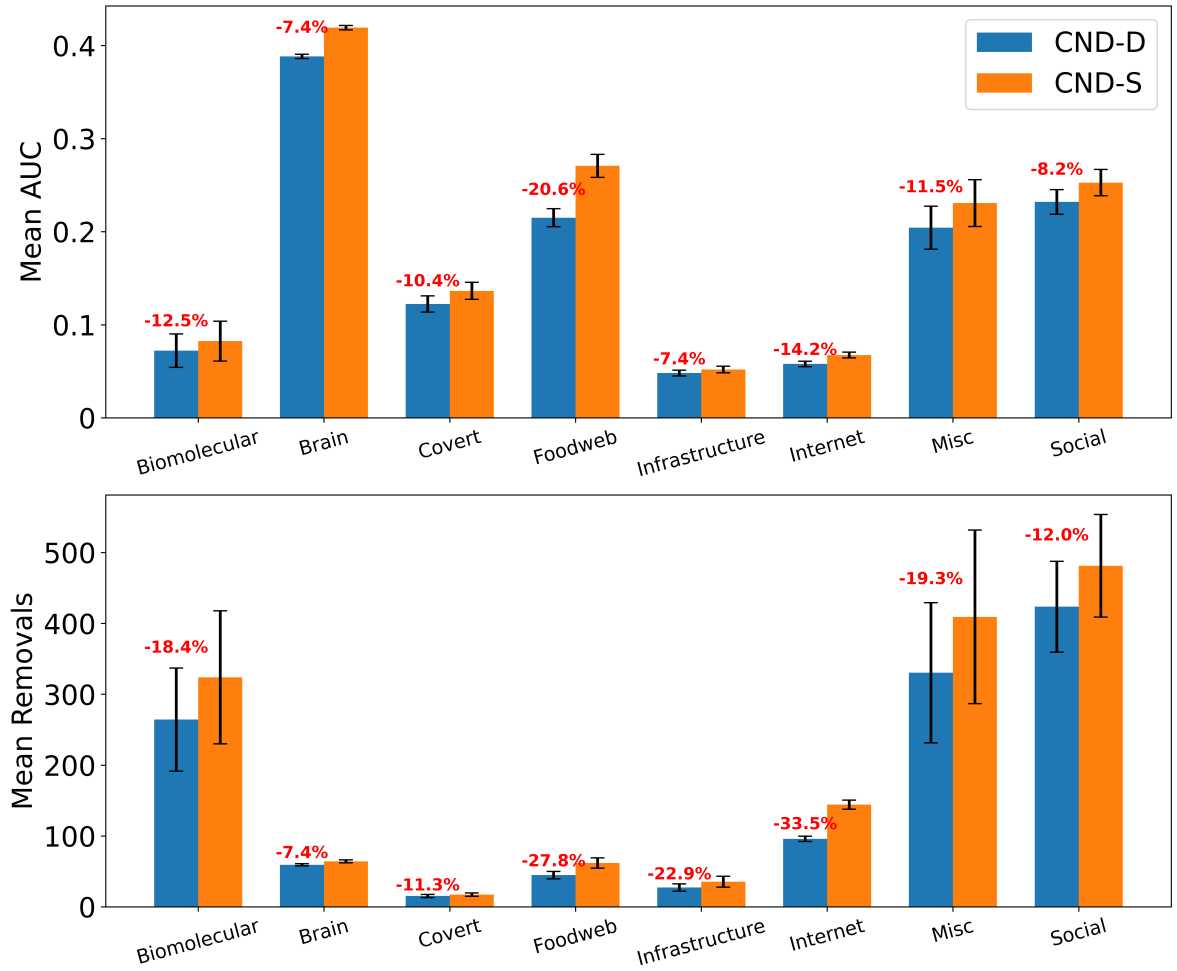

Figure A6. Mean AUC and number of removals for dynamic and static CND (n = 1,296. Error bars represent the standard error of the mean (SEM). Red text indicates the percentage improvement achieved by using dynamic over static variants.

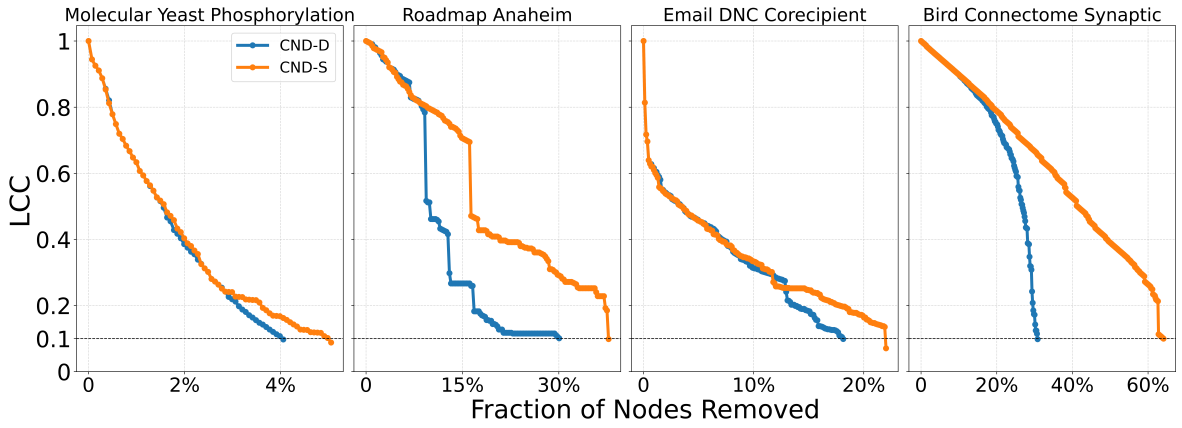

Figure A7. Dismantling process on example networks comparing dynamic and static CND. The plot shows the normalized size of the largest connected component (LCC) as a function of the fraction of nodes removed, with a target LCC threshold of 10%. Performance is evaluated using the Area Under the Curve (AUC) of the LCC trajectory.

Appendix F. Engineering Network Robustness

Here, we would like to comment on the feasibility of our suggested network modifications in practical scenarios. In the case of PPI networks, recent advances in structural modeling of molecules

 doi:10.20944/preprints202505.2276.v2

using AlphaFold 3 (AF3) [50] have reduced the longstanding limitations in testing and engineering arbitrary proteins. Coupling our dismantling predictions with AF3 could impact drug repositioning and drug design. For example, in the case of antibiotic-resistant bacteria, knowledge of how to dismantle the bacterial PPI network could be used to identify which proteins to target with repositioned, modified, or newly designed antibiotics. Meanwhile, to minimize side effects of newly designed drugs that target critical proteins in bacterial PPI networks, our reinforcement strategy based on common-neighbor generation could be applied to predict which protective bindings to promote in the human PPI network, thereby reducing the destructive impact of a drug on a critical human protein whose impairment could cause side effects. Finally, the application of the proposed common-neighbor reinforcement strategy to increase the robustness of flight and shipping networks is straightforward, as it can indicate which critical nodes should be reinforced by adding links between their adjacent nodes.

We validate our reinforcement strategy explained in Section 4.6 on three types of real-world networks considered critical: a human PPI network (biological), a flight map network (transportation with social and geographic constraints), and a shipping trade network (transportation with economic and geographic constraints). We select the top 1% of highest-scoring nodes according to the chosen measure (NBC or CND) and randomly add either 1% or 10% of the potential links between their respective adjacent nodes, thereby modifying the network topology according to the explainable rule discovered via CND.

Table A7. Area Under the dismantling Curve (AUC) for NBC and CND on the original network and "reinforced" networks, by adding 1% and 10% of the links between common neighbours of the top 1% of nodes. In parentheses the increase in the AUC compared to the original network, representing the reduction in the dismantling effectiveness.

Human PPI	0% Links	1% Links	10% Links
NBC-baseline	0.051		
NBC-reinforced		0.093 (+*84*%)	0.159 (+*214*%)
CND-baseline	0.055		
CND-reinforced		0.098 ((+79%))	0.198 ((+259%))

Flight Map US	0% Links	1% Links	10% Links
NBC-baseline	0.042		
NBC-reinforced		0.067 (+*61*%)	0.122 (+*193*%)
CND-baseline	0.055		
CND-reinforced		0.098 ((+95%))	0.172 ((+241%))

Trade Shipping	0% Links	1% Links	10% Links
NBC-baseline	0.138		
NBC-reinforced		0.236 (+*71*%)	0.240 (+*74*%)
CND-baseline	0.220		
CND-reinforced		0.298 ((+36%))	0.350 ((+59%))

 doi:10.20944/preprints202505.2276.v2

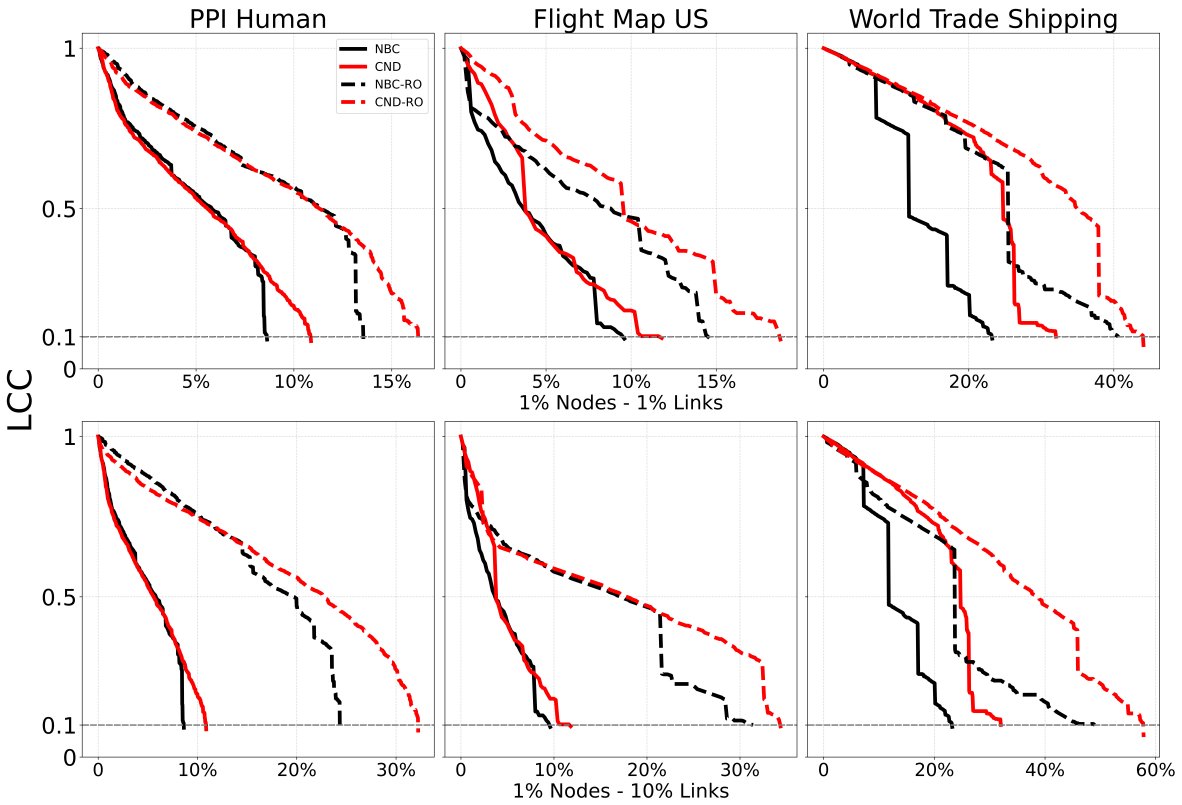

Figure A8. Dismantling curve on the original (solid) and reinforced networks (dashed). The plot shows the normalized size of the largest connected component (LCC) as a function of the fraction of nodes removed, with a target LCC threshold of 10%. Performance is evaluated using the Area Under the Curve (AUC) of the LCC trajectory.

Appendix G. RA Measures in Hyperbolic Networks

Here, we explore the relevance of the RA2 measure as an estimator for latent geometry in hyperbolic networks. As previously mentioned, the RA measures were introduced to serve as pre-weighting strategies for approximating angular distances associated with node similarities in hyperbolic network embeddings [1].

To investigate this, we generate networks using the non-uniform Popularity-Similarity Optimization (nPSO) model [37]. The nPSO model is built on the principle that radial coordinates represent hierarchy (popularity) while angular coordinates represent similarity. It produces networks that are both scale-free (power-law degree distribution, meaning a network has a few highly connected hubs while the majority of nodes have few links) and clustered with distinct communities, closely mimicking the structure of many real-world complex systems.

In Figure A9, we visualize various nPSO networks in the hyperbolic space, keeping the number of nodes ($N = 500$) and communities ($C = 5$) fixed. We test different network topologies by varying:

- The power-law exponent $\gamma \in \{2, 3\}$, which represent common lower and upper bounds for real-world scale-free networks.
- The number of nodes a new node will connect to when being added to the network, $m \in \{5, 10, 20\}$, and represents approximately half of the average node degree, making the network more or less connected.
- The temperature $T \in \{0.3, 0.6, 0.9, 1\}$, which controls the level of clustering (lower temperatures yield stronger clustering).

Nodes in each network are colored according to their RA2 value. The results show that the RA2 measure effectively captures the network's hierarchical structure. Nodes with higher RA2 values are consistently those that are more radially central (i.e., the hubs). This relationship is particularly evident

 doi:10.20944/preprints202505.2276.v2

for $\gamma = 2$, where the skewed degree distribution creates a clearer distinction between central hubs and peripheral nodes. The trend persists for $\gamma = 3$, though it is less pronounced due to the presence of fewer "super-hubs." This confirms that RA2 serves as a reliable proxy for the radial coordinates that encode centrality in hyperbolic network embeddings.

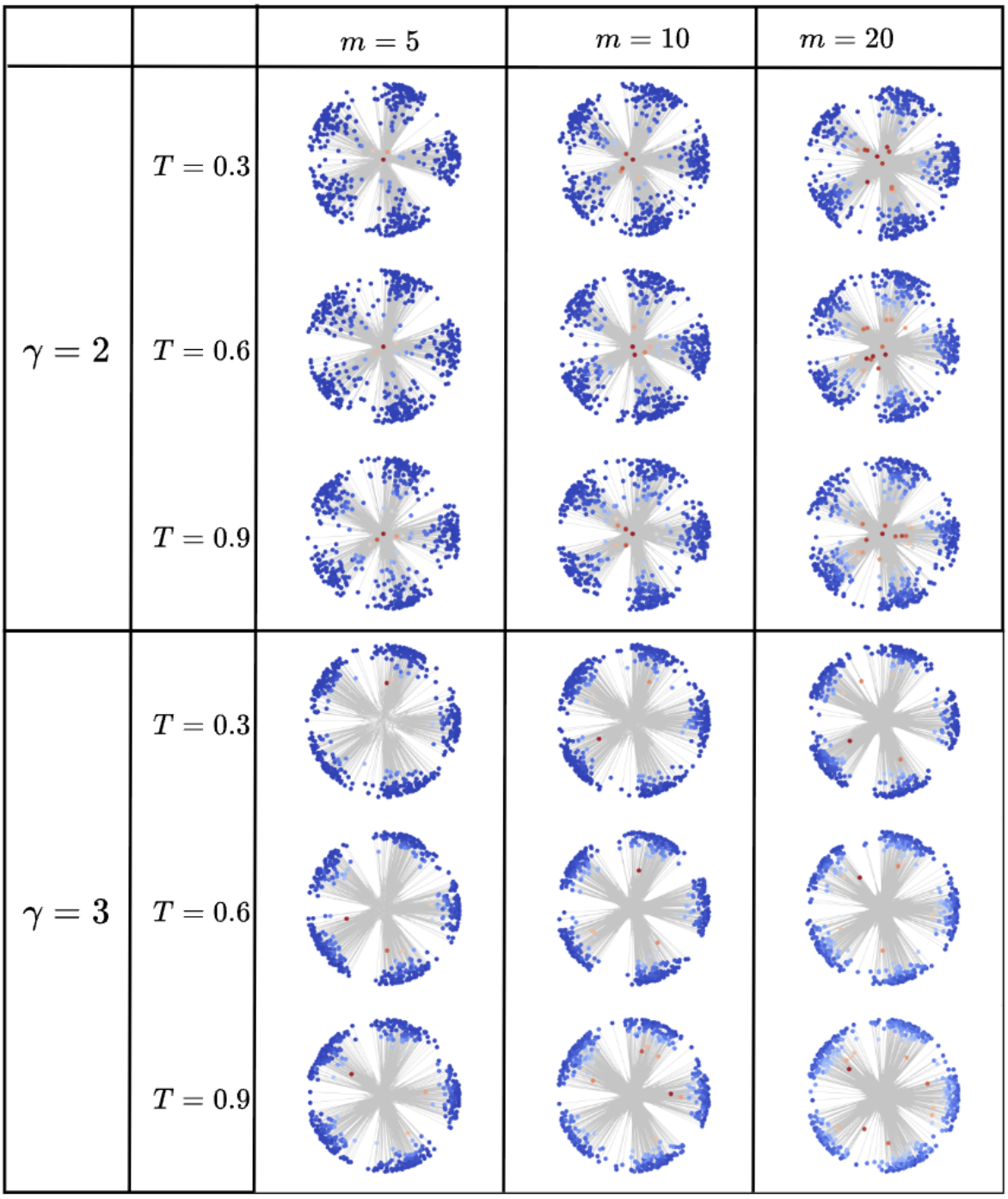

Figure A9. nPSO model networks visualized in the hyperbolic space. Fixed parameters are the number of nodes, N=500, and the number of communities, C=5. Nodes are colored according to their RA2 measure.

 doi:10.20944/preprints202505.2276.v2

Appendix H. GPU Acceleration

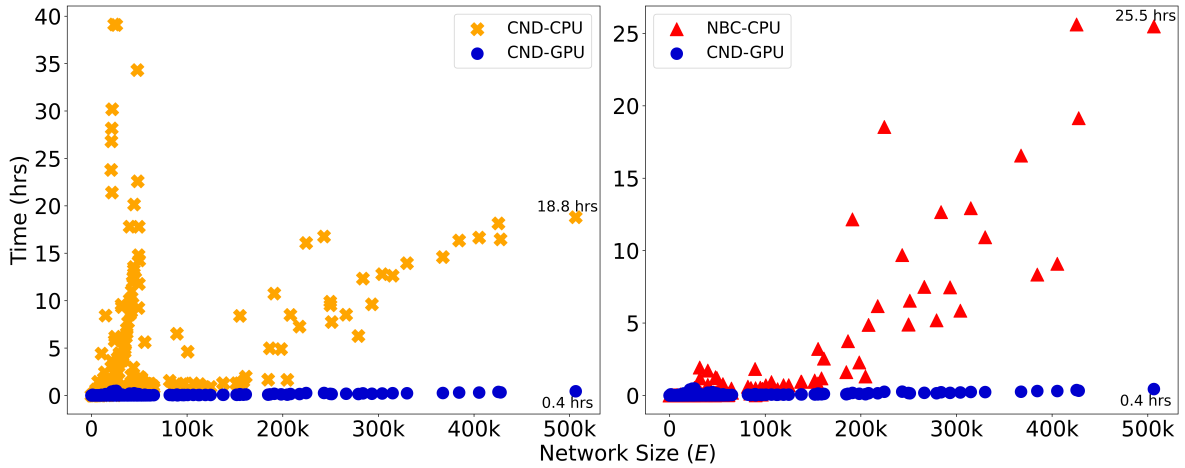

Figure A10. Runtime (in hours) is plotted against network size, measured by the number of edges, E, for dynamic dismantling. The annotated time indicates the runtime for the largest network. Evaluated on networks of up to 23,000 nodes and 507,000 edges ($n = 1{,}475$).

Since the difference in running time between the three LGD-NA methods is not relevant, neither for CPU nor GPU, we report the running time of the original RA2 in the main text (Figure 3) and the CND in the Appendix (Figure A10). When comparing CND and NBC, on the largest network, GPU-accelerated CND is over 46 times faster than its CPU counterpart and also over 63 times faster than NBC running on CPU.

Section 4.5 details how we implement the RA-based measures for GPU, using matrix multiplication. We end up with the following formula for RA2:

$$\mathbf{RA2} = \frac{1 + \mathbf{E}_{\mathrm{L2}} + \mathbf{E}_{\mathrm{L2}}^{\top} + \mathbf{E}_{\mathrm{L2}} \circ \mathbf{E}_{\mathrm{L2}}^{\top}}{1 + \mathbf{CN}_{\mathrm{L2}}}$$

and for CND:

$$\mathbf{CND} = \frac{1}{1 + \mathbf{CN}_{\mathrm{L2}}}$$

In our experiments, the CPU-based RA2 and CND implementation uses Python's `NumPy` (implemented in C) while the GPU implementation uses Python's `CuPy` (implemented in C++/CUDA).

As mentioned earlier, we report only the CPU running time for NBC, as its GPU implementation did not yield any speedup. While some studies report GPU implementations of NBC with improved performance [51–56], these are often limited by hardware-specific optimizations, data-specific assumptions (e.g., small-world, social, or biological networks), and using heuristics that are tailored to specific settings rather than offering general solutions. Moreover, publicly available code is rare, making these approaches difficult to reproduce or integrate. Overall, NBC is not naturally suited for GPU implementation, as it does not rely on matrix multiplication, but is based on computing shortest path counts between all node pairs. In our experiments, the CPU-based NBC implementation from Python's `graph_tool` (implemented in C++), based on Brandes' algorithm [45] with time complexity $\mathcal{O}(Nm)$ for unweighted graphs, outperformed the GPU version from Python's `cuGraph` (implemented for C++/CUDA).

Appendix I. Experimental Setup

Baseline Topological Centrality Measures.

We selected centrality measures to cover diverse categories: shortest path-based (NBC), degree-based (degree), walk-based (eigenvector), random walk-based (PageRank), resilience-based (Re-

 doi:10.20944/preprints202505.2276.v2

silience), and fitness-based (Domirank and Fitness centrality). We also tested closeness and load centrality, but both performed worse than NBC and rely on the same shortest-path principle; thus, we retained NBC. Similarly, Katz centrality underperformed compared to eigenvector centrality and is also based on spectral properties of the adjacency matrix. For DomiRank, we tested three values for the numerator in the σ parameter formula: 0.1, 0.5, and 0.9. While the original study sometimes performs a grid search to find the optimal σ per network, this is not feasible for our large-scale evaluation. Instead, we selected a representative range and found that $\sigma_{num} = 0.5$ yielded the best performance, and we report that value. The parameter σ controls the balance between local degree-based dominance and global network-structure-based competition. As $\sigma \to 0$, the scores approximate degree centrality. As σ increases, nodes are evaluated in increasingly competitive environments, where centrality depends more on non-local structural dominance than individual connectivity. For Fitness centrality, we capped the number of iterations at $k = 100$. Without this cap, the method took a prohibitively long time to converge, especially on large networks.

Baseline Dismantling Methods.

We selected the best-performing and most widely adopted dismantling algorithms from the literature [6]. As mentioned earlier, we did not include BPD and FINDER in our experiments due to unavailable and outdated code, respectively. For Collective Influence (CI), we tested values $\ell = 1, 2, 3$, where ℓ defines the radius of a ball centered at node i, and $\partial B(i, \ell)$ is the frontier at distance ℓ (i.e., nodes exactly ℓ hops away). We found that $\ell = 1$ performed best across our benchmarks and report this setting, while $\ell = 3$ performed the worst. For Explosive Immunization (EI), we evaluated both scores $\sigma^{(1)}$ and $\sigma^{(2)}$. The $\sigma^{(1)}$ score targets high-connectivity nodes to rapidly fragment the network early on. The $\sigma^{(2)}$ score aims to prevent large cluster merges near the percolation threshold by avoiding the connection of separate components. We found that $\sigma^{(1)}$ consistently outperformed $\sigma^{(2)}$, and thus we use it in our final experiments. For eigenvector centrality, we capped the number of power iterations at $k = 100$ to avoid long or unbounded runtimes, since convergence can be very slow in large networks. For PageRank, we used a convergence tolerance of $\epsilon = 10^{-6}$, as the algorithm runs until the change in scores falls below this threshold.

Note on Reinsertion.

We did not apply reinsertion techniques on all networks included in the initial dismantling experiments. In some cases, certain methods performed so poorly that applying reinsertion became prohibitively slow. To ensure consistency, we excluded these networks from the reinsertion analysis for all methods. Specifically, we imposed a cutoff: networks were excluded if any method required more than 800 node removals to reach the dismantling threshold. Based on this criterion, 59 networks were excluded.

Note on Evaluation.

A lower number of removals does not always imply a lower AUC. Our AUC metric rewards methods that fragment the network early, even if they require more steps to reach the dismantling threshold. As shown in Figure A11, we show cases where a method that reaches the threshold with more removals can still achieve a lower AUC, due to earlier damage to the network structure.

 doi:10.20944/preprints202505.2276.v2

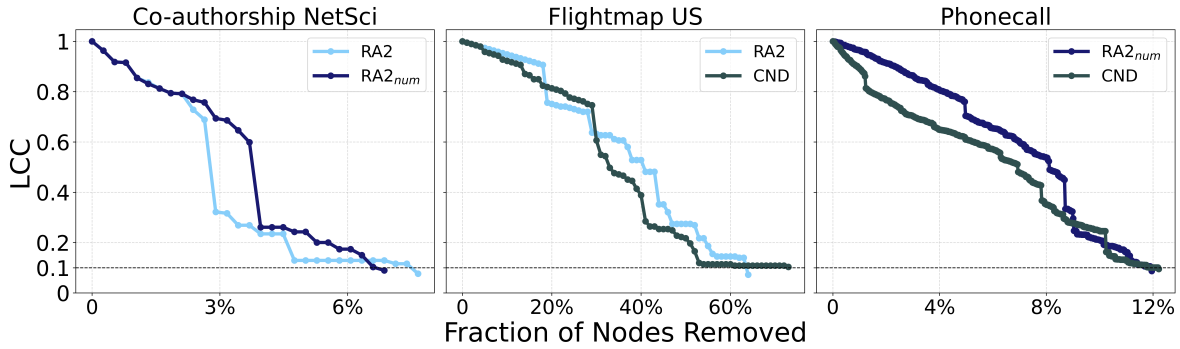

Figure A11. Dynamic dismantling process on example networks comparing CND, RA2 and RA2$_{num}$, showing a lower removal number does not necessarily mean a lower Area Under the Curve (AUC) of the LCC trajectory. The plot shows the normalized size of the largest connected component (LCC) as a function of the fraction of nodes removed, with a target LCC threshold of 10%.

Code.

All our methods are implemented in our codebase, available in the supplementary material. We implement RA2, RA2$_{num}$, CND, NBC, as well as degree and eigenvector centrality. We also adapt and integrate the original implementations of Domirank centrality and Fitness centrality. Since no code was available for Resilience centrality, we implemented it ourselves based on the original description in the paper. Instructions for running these methods and reproducing the experiments are included in the supplementary material. We also provide a representative network from the ATLAS dataset for testing purposes. For GDM, CoreGDM, GND, CI, EI, MS, and CoreHD, we use the publicly available code from Artime et al. (2024)'s review.

Computational Resources.

All experiments were conducted on a machine equipped with an AMD Ryzen Threadripper PRO 3995WX CPU (64 cores), 251 GiB of RAM, and a single NVIDIA RTX A4000 GPU with 16 GiB of memory. All code was implemented in Python, with dependencies and library versions specified in the supplementary material to ensure full reproducibility.

Quantitative Results.

All results are in Tables S1–S13 in the Supplementary Material.

Appendix J. Discussion, Limitations, and Future Work

Missing or Manipulated Data.

The robustness of our LGD-NA methods to missing or manipulated information, such as missing neighbor data or adversarial modifications to neighborhood structure, is a crucial question for practical applications of any network analysis method, especially those relying on local information. Our LGD-NA methods, by design, are inherently more robust to global missing or manipulated information compared to methods that require a complete global view of the network (e.g., exact NBC). Since our methods rely solely on local neighborhood information, random missing data across the network would primarily affect only the scores of directly impacted nodes, rather than propagating errors throughout the entire graph. Similarly, adversarial modifications would need to target specific local neighborhoods to significantly alter a node's dismantling score, making large-scale, coordinated attacks more challenging than for global metrics. However, we acknowledge that direct adversarial manipulation of a node's immediate neighborhood could indeed impact its calculated score.

NBC Approximators

While the prohibitive computational cost of the highly effective Node Betweenness Centrality (NBC) metric, a global estimator of latent geometry, has led to the development of numerous approxi-

 doi:10.20944/preprints202505.2276.v2

mators [57–60], our proposed LGD-NA methods are founded on a fundamentally different principle. Existing NBC approximators primarily focus on creating faster implementations of a global metric. In contrast, our approach leverages purely local topological network information to directly estimate pairwise distances in the latent metric space. A crucial distinction is that our method bypasses the need for the global knowledge or complex sampling strategies upon which many NBC approximators still depend. Although these approximation techniques can be faster than calculating the exact NBC, their reliance on sampling or broader computations is inherently less efficient than our strictly local approach. Furthermore, the imperfections introduced by sampling can degrade dismantling performance. Such sampling of global information is also not robust to missing data and remains vulnerable to adversarial attacks. The strength of our method, therefore, lies in its ability to achieve high dismantling performance by directly utilizing local geometric insights, rather than attempting to approximate a computationally intensive, global metric like NBC.

Limitations.

A limitation of this study is the mismatch between theoretical and observed runtimes, which can vary across methods. These differences stem from factors such as the programming language used, hardware acceleration, and implementation-level optimizations. However, all experiments were run on the same CPU and GPU to ensure a fair comparison, and we made a strong effort to optimize all methods both in terms of runtime and dismantling performance. For example, we tested different parameters for Domirank, CI, and EI, and evaluated multiple variants of shortest path-based and spectral partitioning-based centrality measures. Furthermore, we were unable to test on extremely large networks due to hardware constraints and the high computational cost of running a broad set of dismantling methods. However, we are confident that our results would generalize to larger networks, given the diversity of the 1,475 networks tested, spanning a wide range of domains and sizes from very small to large. Another limitation relates to the parameter tuning required by some baseline methods, especially machine learning-based approaches and Domirank. Due to the scale of our experimental setup—both in the number and size of networks—we were unable to perform extensive tuning. Although targeted tuning could enhance performance for specific methods on individual networks, it would compromise consistency across the wide range of complex systems domains considered. In contrast, LGD-NA requires no parameter tuning and consistently achieves strong, generalizable performance across all tested networks.

Future Work.

Future research could further explore latent geometry, particularly how to effectively combine local and global information in dismantling strategies. Improving the scalability of matrix-based computations, especially for very large and sparse networks, is another important direction. There is also a need for more cost-efficient dynamic dismantling strategies that reduce the overhead of recomputing scores after every node removal without significantly sacrificing performance. In addition, edge dismantling remains a relatively underexplored area compared to node-based dismantling, and it would be valuable to investigate whether latent geometry-driven principles can also guide the efficient removal of links in complex networks. Targeting edges can be just as important as targeting nodes, and in many real-world systems, such as transportation networks (railroads, roads, subways, or shipping trade routes), edge removal may represent the more realistic and sensible threat scenario, making it highly relevant for dismantling strategies. Finally, our work can also be of interest to Explainable AI (XAI) for models like GNNs and Reservoir Computers (RC). By identifying critical (dismantling) or unimportant (percolation) nodes in a network, we can conduct "perturbation experiments" to assess their impact on an ML model's performance. This process reveals which parts of the graph are most crucial for the model's predictions or computations, effectively opening the model's "black box" and providing crucial insights into its internal decision-making, robustness, and vulnerabilities.

 doi:10.20944/preprints202505.2276.v2

Appendix K. Ethics Statement

The research on network dismantling presented in this paper has a potential for dual use. The techniques developed to identify and exploit network vulnerabilities could theoretically be used to design targeted attacks on critical systems, such as communication, transportation, or power grid networks. However, it can also be used for defensive strategies. A thorough understanding of network vulnerabilities is a prerequisite for designing robust systems. To directly address the dual-use concern, we demonstrate the constructive potential of our work by presenting a novel technique for proactively engineering network robustness using our latent geometry-based methods (see Section 4.6). This application moves beyond vulnerability analysis to provide an explainable framework for modifying a network to enhance its resilience. Finally, by openly publishing our theoretical foundations, source code, and comprehensive evaluations, we aim to ensure that the benefits of this research, namely the ability to secure critical networks, are accessible to all. We believe the societal benefit of advancing defensive capabilities significantly outweighs the risk of potential misuse.

Appendix L. Reproducibility Statement

To ensure full reproducibility, we have made our source code publicly available, including detailed instructions on how to replicate all experiments. The codebase includes an implementation of our LGD-NA framework (illustrated in Figure 1), the exact formulas used (detailed in Appendix A), and an example network for demonstration. The code is compatible with both CPU and GPU environments and also provides the necessary tools to engineer network robustness as described in this work. The baseline methods were implemented using the code from the review by Artime et al. [6]. The exact topological measures of all networks used in our study are provided in Appendix A5. Further details regarding the experimental setup, including hardware specifications, are described in Appendix I.

Appendix M. Claim of the LLM usage

We used LLM-based tools to improve the language and flow; the principles, core logic, and innovations are entirely the authors'.

References

1. Muscoloni, A.; Thomas, J.M.; Ciucci, S.; Bianconi, G.; Cannistraci, C.V. Machine learning meets complex networks via coalescent embedding in the hyperbolic space. *Nature Communications* **2017**, *8*. https://doi.org/10.1038/s41467-017-01825-5.
2. Braunstein, A.; Dall'Asta, L.; Semerjian, G.; Zdeborová, L. Network dismantling. *Proceedings of the National Academy of Sciences* **2016**, *113*, 12368–12373, [https://www.pnas.org/doi/pdf/10.1073/pnas.1605083113]. https://doi.org/10.1073/pnas.1605083113.
3. Morone, F.; Min, B.; Bo, L.; Mari, R.; Makse, H.A. Collective Influence Algorithm to find influencers via optimal percolation in massively large social media. *Scientific Reports* **2016**, *6*. https://doi.org/10.1038/srep30062.
4. Mugisha, S.; Zhou, H.J. Identifying optimal targets of network attack by belief propagation. *Phys. Rev. E* **2016**, *94*, 012305. https://doi.org/10.1103/PhysRevE.94.012305.
5. Newman, M.E.J. The Structure and Function of Complex Networks. *SIAM Review* **2003**, *45*, 167–256, [https://doi.org/10.1137/S003614450342480]. https://doi.org/10.1137/S003614450342480.
6. Artime, O.; Grassia, M.; De Domenico, M.; Gleeson, J.P.; Makse, H.A.; Mangioni, G.; Perc, M.; Radicchi, F. Robustness and resilience of complex networks. *Nature Reviews Physics* **2024**, *6*. https://doi.org/10.1038/s42254-023-00676-y.
7. Albert, R.; Jeong, H.; Barabási, A.L. Error and attack tolerance of complex networks. *Nature* **2000**, *406*, 378–382. https://doi.org/10.1038/35019019.
8. Barabási, A.L.; Albert, R. Emergence of Scaling in Random Networks. *Science* **1999**, *286*, 509–512, [https://www.science.org/doi/pdf/10.1126/science.286.5439.509]. https://doi.org/10.1126/science.286.5439.509.
9. Broido, A.D.; Clauset, A. Scale-free networks are rare. *Nature Communications* **2019**, *10*, 1017. https://doi.org/10.1038/s41467-019-08746-5.

 doi:10.20944/preprints202505.2276.v2

10. Voitalov, I.; van der Hoorn, P.; van der Hofstad, R.; Krioukov, D. Scale-free networks well done. *Phys. Rev. Res.* **2019**, *1*, 033034. https://doi.org/10.1103/PhysRevResearch.1.033034.
11. Serafino, M.; Cimini, G.; Maritan, A.; Rinaldo, A.; Suweis, S.; Banavar, J.R.; Caldarelli, G. True scale-free networks hidden by finite size effects. *Proceedings of the National Academy of Sciences* **2021**, *118*, e2013825118, [https://www.pnas.org/doi/pdf/10.1073/pnas.2013825118]. https://doi.org/10.1073/pnas.2013825118.
12. Newman, M.E.J. Communities, modules and large-scale structure in networks. *Nature Physics* **2012**, *8*, 25–31. https://doi.org/10.1038/nphys2162.
13. Ravasz, E.; Barabási, A.L. Hierarchical organization in complex networks. *Phys. Rev. E* **2003**, *67*, 026112. https://doi.org/10.1103/PhysRevE.67.026112.
14. Clauset, A.; Moore, C.; Newman, M.E.J. Hierarchical structure and the prediction of missing links in networks. *Nature* **2008**, *453*, 98–101. https://doi.org/10.1038/nature06830.
15. Battiston, F.; Amico, E.; Barrat, A.; Bianconi, G.; Ferraz de Arruda, G.; Franceschiello, B.; Iacopini, I.; Kéfi, S.; Latora, V.; Moreno, Y.; et al. The physics of higher-order interactions in complex systems. *Nature Physics* **2021**, *17*, 1093–1098. https://doi.org/10.1038/s41567-021-01371-4.
16. Lambiotte, R.; Rosvall, M.; Scholtes, I. From networks to optimal higher-order models of complex systems. *Nature Physics* **2019**, *15*, 313–320. https://doi.org/10.1038/s41567-019-0459-y.
17. Wu, Z.; Menichetti, G.; Rahmede, C.; Bianconi, G. Emergent Complex Network Geometry. *Scientific Reports* **2015**, *5*, 10073. https://doi.org/10.1038/srep10073.
18. Boguñá, M.; Bonamassa, I.; De Domenico, M.; Havlin, S.; Krioukov, D.; Serrano, M.A. Network geometry. *Nature Reviews Physics* **2021**, *3*. https://doi.org/10.1038/s42254-020-00264-4.
19. Krioukov, D.; Papadopoulos, F.; Kitsak, M.; Vahdat, A.; Boguñá, M. Hyperbolic geometry of complex networks. *Phys. Rev. E* **2010**, *82*, 036106. https://doi.org/10.1103/PhysRevE.82.036106.
20. Serrano, M.A.; Krioukov, D.; Boguñá, M. Self-Similarity of Complex Networks and Hidden Metric Spaces. *Phys. Rev. Lett.* **2008**, *100*, 078701. https://doi.org/10.1103/PhysRevLett.100.078701.
21. Freeman, L.C. A Set of Measures of Centrality Based on Betweenness. *Sociometry* **1977**, *40*, 35–41.
22. Engsig, M.; Tejedor, A.; Moreno, Y.; Foufoula-Georgiou, E.; Kasmi, C. DomiRank Centrality reveals structural fragility of complex networks via node dominance. *Nature Communications* **2024**, *15*, 56. https://doi.org/10.1038/s41467-023-44257-0.
23. Servedio, V.D.P.; Bellina, A.; Calò, E.; De Marzo, G. Fitness centrality: a non-linear centrality measure for complex networks. *Journal of Physics: Complexity* **2025**, *6*, 015002. https://doi.org/10.1088/2632-072X/ada845.
24. Motter, A.E.; Lai, Y.C. Cascade-based attacks on complex networks. *Phys. Rev. E* **2002**, *66*, 065102. https://doi.org/10.1103/PhysRevE.66.065102.
25. Holme, P.; Kim, B.J.; Yoon, C.N.; Han, S.K. Attack vulnerability of complex networks. *Phys. Rev. E* **2002**, *65*, 056109. https://doi.org/10.1103/PhysRevE.65.056109.
26. Boguñá, M.; Krioukov, D.; Claffy, K.C. Navigability of complex networks. *Nature Physics* **2009**, *5*. https://doi.org/10.1038/nphys1130.
27. Kleinberg, J.M. Navigation in a small world. *Nature* **2000**, *406*. https://doi.org/10.1038/35022643.
28. Muscoloni, A.; Cannistraci, C.V. Navigability evaluation of complex networks by greedy routing efficiency. *Proceedings of the National Academy of Sciences* **2019**, *116*, 1468–1469, [https://www.pnas.org/doi/pdf/10.1073/pnas.1817880116]. https://doi.org/10.1073/pnas.1817880116.
29. Muscoloni, A.; Cannistraci, C.V. Leveraging the nonuniform PSO network model as a benchmark for performance evaluation in community detection and link prediction. *New Journal of Physics* **2018**, *20*, 063022. https://doi.org/10.1088/1367-2630/aac6f9.
30. Zdeborová, L.; Zhang, P.; Zhou, H.J. Fast and simple decycling and dismantling of networks. *Scientific Reports* **2016**, *6*. https://doi.org/10.1038/srep37954.
31. Clusella, P.; Grassberger, P.; Pérez-Reche, F.J.; Politi, A. Immunization and Targeted Destruction of Networks using Explosive Percolation. *Phys. Rev. Lett.* **2016**, *117*, 208301. https://doi.org/10.1103/PhysRevLett.117.208301.
32. Ren, X.L.; Gleinig, N.; Helbing, D.; Antulov-Fantulin, N. Generalized network dismantling. *Proceedings of the National Academy of Sciences* **2019**, *116*, 6554–6559, [https://www.pnas.org/doi/pdf/10.1073/pnas.1806108116]. https://doi.org/10.1073/pnas.1806108116.
33. Zhang, Y.; Shao, C.; He, S.; Gao, J. Resilience centrality in complex networks. *Phys. Rev. E* **2020**, *101*, 022304. https://doi.org/10.1103/PhysRevE.101.022304.

 doi:10.20944/preprints202505.2276.v2

34. Grassia, M.; De Domenico, M.; Mangioni, G. Machine learning dismantling and early-warning signals of disintegration in complex systems. *Nature Communications* **2021**, *12*. https://doi.org/10.1038/s41467-021-25485-8.
35. Grassia, M.; Mangioni, G. CoreGDM: Geometric Deep Learning Network Decycling and Dismantling. In Proceedings of the Complex Networks XIV; Teixeira, A.S.; Botta, F.; Mendes, J.F.; Menezes, R.; Mangioni, G., Eds., Cham, 2023; pp. 86–94.
36. Zuev, K.; Boguñá, M.; Bianconi, G.; Krioukov, D. Emergence of Soft Communities from Geometric Preferential Attachment. *Scientific Reports* **2015**, *5*, 9421. https://doi.org/10.1038/srep09421.
37. Muscoloni, A.; Cannistraci, C.V. A nonuniform popularity-similarity optimization (nPSO) model to efficiently generate realistic complex networks with communities. *New Journal of Physics* **2018**, *20*, 052002. https://doi.org/10.1088/1367-2630/aac06f.
38. Papadopoulos, F.; Kitsak, M.; Serrano, M.Á.; Boguñá, M.; Krioukov, D. Popularity versus similarity in growing networks. *Nature* **2012**, *489*, 537–540. https://doi.org/10.1038/nature11459.
39. Brockmann, D.; Helbing, D. The Hidden Geometry of Complex, Network-Driven Contagion Phenomena. *Science* **2013**, *342*, 1337–1342, [https://www.science.org/doi/pdf/10.1126/science.1245200]. https://doi.org/10.1126/science.1245200.
40. Bonacich, P. Factoring and weighting approaches to status scores and clique identification. *The Journal of Mathematical Sociology* **1972**, *2*, 113–120, [https://doi.org/10.1080/0022250X.1972.9989806]. https://doi.org/10.1080/0022250X.1972.9989806.
41. Page, L.; Brin, S.; Motwani, R.; Winograd, T. The PageRank Citation Ranking: Bringing Order to the Web. Technical Report 1999-66, Stanford InfoLab, 1999. Previous number = SIDL-WP-1999-0120.
42. Fan, C.; Zeng, L.; Sun, Y.; Liu, Y.Y. Finding key players in complex networks through deep reinforcement learning. *Nature Machine Intelligence* **2020**, *2*, 317–324. https://doi.org/10.1038/s42256-020-0177-2.
43. Bianconi, G.; Darst, R.K.; Iacovacci, J.; Fortunato, S. Triadic closure as a basic generating mechanism of communities in complex networks. *Phys. Rev. E* **2014**, *90*, 042806. https://doi.org/10.1103/PhysRevE.90.042806.
44. Fan, C.; Zeng, L.; Feng, Y.; Xiu, B.; Huang, J.; Liu, Z. Revisiting the power of reinsertion for optimal targets of network attack. *Journal of Cloud Computing* **2020**, *9*. https://doi.org/10.1186/s13677-020-00169-8.
45. Brandes, U. A faster algorithm for betweenness centrality*. *The Journal of Mathematical Sociology* **2001**, *25*, 163–177, [https://doi.org/10.1080/0022250X.2001.9990249]. https://doi.org/10.1080/0022250X.2001.9990249.
46. Schneider, C.M.; Mihaljev, T.; Herrmann, H.J. Inverse targeting —An effective immunization strategy. *Europhysics Letters* **2012**, *98*, 46002. https://doi.org/10.1209/0295-5075/98/46002.
47. Newman, M.E.J. Assortative Mixing in Networks. *Phys. Rev. Lett.* **2002**, *89*, 208701. https://doi.org/10.1103/PhysRevLett.89.208701.
48. Cannistraci, C.V.; Alanis-Lobato, G.; Ravasi, T. From link-prediction in brain connectomes and protein interactomes to the local-community-paradigm in complex networks. *Scientific Reports* **2013**, *3*, 1613. https://doi.org/10.1038/srep01613.
49. Newman, M.E.J. Detecting community structure in networks. *The European Physical Journal B* **2004**, *38*, 321–330. https://doi.org/10.1140/epjb/e2004-00124-y.
50. Abramson, J.; Adler, J.; Dunger, J.; Evans, R.; Green, T.; Pritzel, A.; Ronneberger, O.; Willmore, L.; Ballard, A.J.; Bambrick, J.; et al. Accurate structure prediction of biomolecular interactions with AlphaFold 3. *Nature* **2024**, *630*, 493–500. https://doi.org/10.1038/s41586-024-07487-w.
51. Fan, R.; Xu, K.; Zhao, J. A GPU-based Solution for Fast Calculation of the Betweenness Centrality in Large Weighted Networks. *PeerJ Computer Science* **2017**, *3*, e140. https://doi.org/10.7717/peerj-cs.140.
52. Shi, Z.; Zhang, B. Fast network centrality analysis using GPUs. *BMC Bioinformatics* **2011**, *12*, 149. https://doi.org/10.1186/1471-2105-12-149.
53. Pande, P.; Bader, D.A. Computing Betweenness Centrality for Small World Networks on a GPU. In Proceedings of the Proceedings of the 15th Annual High Performance Embedded Computing Workshop (HPEC), September 2011.
54. McLaughlin, A.; Bader, D.A. Accelerating GPU betweenness centrality. *Commun. ACM* **2018**, *61*, 85–92. https://doi.org/10.1145/3230485.
55. Sariyüce, A.E.; Kaya, K.; Saule, E.; Çatalyürek, U.V. Betweenness centrality on GPUs and heterogeneous architectures. In Proceedings of the Proceedings of the 6th Workshop on General Purpose Processor Using Graphics Processing Units, New York, NY, USA, 2013; GPGPU-6, p. 76–85. https://doi.org/10.1145/2458523.2458531.

 doi:10.20944/preprints202505.2276.v2

56. Bernaschi, M.; Carbone, G.; Vella, F. Scalable betweenness centrality on multi-GPU systems. In Proceedings of the Proceedings of the ACM International Conference on Computing Frontiers, New York, NY, USA, 2016; CF '16, p. 29–36. https://doi.org/10.1145/2903150.2903153.
57. Bader, D.A.; Kintali, S.; Madduri, K.; Mihail, M. Approximating Betweenness Centrality. In Proceedings of the Algorithms and Models for the Web-Graph; Bonato, A.; Chung, F.R.K., Eds., Berlin, Heidelberg, 2007; pp. 124–137.
58. Bergamini, E.; Meyerhenke, H. Fully-Dynamic Approximation of Betweenness Centrality. In Proceedings of the Algorithms - ESA 2015; Bansal, N.; Finocchi, I., Eds., Berlin, Heidelberg, 2015; pp. 155–166.
59. Haghir Chehreghani, M. An efficient algorithm for approximate betweenness centrality computation. In Proceedings of the Proceedings of the 22nd ACM International Conference on Information & Knowledge Management, New York, NY, USA, 2013; CIKM '13, p. 1489–1492. https://doi.org/10.1145/2505515.2507826.
60. Riondato, M.; Kornaropoulos, E.M. Fast approximation of betweenness centrality through sampling. In Proceedings of the Proceedings of the 7th ACM International Conference on Web Search and Data Mining, New York, NY, USA, 2014; WSDM '14, p. 413–422. https://doi.org/10.1145/2556195.2556224.