EXACT–LEARNED DOOB CONTROL FOR NETWORK DISMANTLING UNDER STRUCTURAL SHIFT

Anonymous authors
Paper under double-blind review

ABSTRACT

Learned network dismantling methods can struggle to generalize to networks whose structures differ from those seen during training. To address this limitation, we separate the effect of removing a node into two components: an immediate effect that can be computed exactly and a budget-dependent future effect that must be estimated. We formulate network dismantling as an irreversible stochastic process over nested residual graphs and use a discrete Doob transform to steer a topology-based reference process toward early fragmentation. The immediate effect of each candidate node is computed using a linear-time articulation analysis, while a neural network estimates only the remaining long-term contribution. The model is trained using exact dynamic programming on small graphs together with a self-consistency objective on larger unlabeled graphs.

Our proposed method, ELDOR, is trained only on small synthetic networks and generalizes zero-shot to 48 real-world networks and a standard synthetic benchmark. ELDOR outperforms the strongest learned baseline on 42 of the 48 real networks and reduces mean normalized AUC by 6.8%. Its overall performance is comparable to strong training-free methods, while providing significant improvements on biological, information, and social networks. Using best-of-16 search further improves ELDOR's average performance across the benchmark.

1 INTRODUCTION

Network dismantling asks for an ordered set of node removals that breaks a network into small components as quickly as possible (Albert et al., 2000; Cohen et al., 2001; Braunstein et al., 2016). Its solutions locate the nodes on which the connectivity of a networked system depends, which matters for infrastructure robustness, targeted immunization and protection against cascading disconnection (Schneider et al., 2011; Artime et al., 2024). The problem is sequential: on the US power grid, two adaptive removal rules that spend the same budget of 1% of the nodes leave either 94% or 24% of the grid connected, because each rule chooses its next node on the graph left by its earlier choices, and the area under their dismantling curves differs more than fourfold (Figure 1).

Since dismantling is NP-hard (Braunstein et al., 2016; Arulselvan et al., 2009), only approximate solutions can be sought. Early solutions rank nodes by centrality (Freeman, 1977), later ones solve tractable proxies such as optimal percolation, decycling, core decomposition and spectral partitioning (Morone & Makse, 2015; Braunstein et al., 2016; Zdeborová et al., 2016; Ren et al., 2019), and recomputed latent-geometry scores, LGD-NA and its rule CND, set a strong training-free standard (Adler et al., 2026). Learned dismantlers instead train graph neural networks on small synthetic networks and deploy them on larger ones (Fan et al., 2020; Grassia et al., 2021; Zhang & Wang, 2022; Tian et al., 2026; Qu et al., 2025; Hong et al., 2025). Reinforcement-learning dismantlers such as FINDER and MIND construct removal trajectories, and GRLND selects a removal set in a single step, yet each of them learns the whole value of a removal, immediate and future effects together, from message-passing features. Ranking methods such as GDM do not model the order of removals at all.

We argue that the difficulty lies in what the network is asked to learn. The value of a removal has two parts. Its immediate effect on the giant component can be computed exactly in linear time, yet it is invisible to message-passing networks, which cannot detect cut vertices (Zhang et al., 2023). Its effect on what the remaining budget can still achieve is, in turn, precisely what ranking formulations

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

leave out. A network that must learn both from data is pushed toward correlates such as degree, which hold in the training families but fail under structural shift (Yehudai et al., 2021) and on separators that work only when removed together. How much of the value of a removal can then be computed rather than learned, and what must the learned remainder capture for a whole trajectory to transfer?

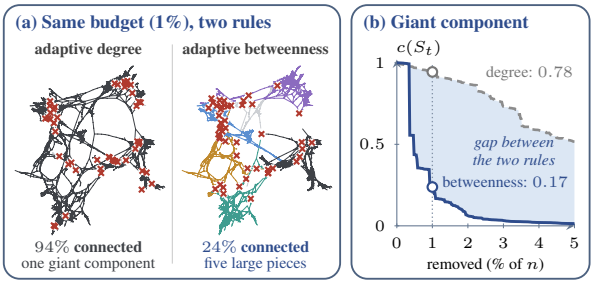

Figure 1: **The removal rule decides fragmentation.** (a) US power grid after the same 1% budget under adaptive degree or adaptive betweenness. (b) Largest component over the first 5%, with the area under each curve; the shaded gap is their difference.

To address this, we propose ELDOR (Exact–Learned Doob-controlled Removal), which computes the first part exactly and learns only the second. ELDOR models dismantling as a stochastic process over irreversible removal trajectories that evolves through nested sets of surviving nodes $V = S_0 \supseteq S_1 \supseteq \cdots \supseteq S_B$, removing one node per transition. Specifically, i) a topology-only reference kernel proposes removals; ii) a budget-conditioned potential reweights them through a discrete Doob transform (Doob, 1957), so that with the exact potential the process samples a path measure that favors early fragmentation (Del Moral, 2004; Todorov, 2009), i.e., KL-regularized control with soft afterstate values (Sutton & Barto, 2018; Haarnoja et al., 2017); and iii) a structural clock computed from exact articulation sensitivities replaces the uniform time index and concentrates computation where the network is about to fragment. The potential is learned from exact dynamic programming on small graphs and from consistency objectives on larger unlabeled graphs. Our main contributions are:

- **A controlled formulation.** We cast dismantling as a Doob-controlled removal process whose exact potential samples a path law tilted toward early fragmentation (Theorem 2); validity and permutation equivariance hold for every parameter value (Propositions 3 and 4).
- **An exact–learned split.** We compute the afterstate cost of every candidate exactly by a linear-time articulation pass (Lemma 1) and learn only the budget-dependent future term with a structure-only network, using exact labels on small graphs alone.
- **A learning guarantee.** We show that a potential within ϵ of the exact one yields a trajectory law within $2\epsilon B$ in KL (Proposition 8); we check this premise exactly up to $n = 20$ and, beyond, report only consistency diagnostics (Table 17).
- **A zero-shot evaluation.** We evaluate one model, trained only on synthetic graphs, on the 48 real networks and a synthetic test protocol against nine published methods and six further baselines, reporting greedy and best-of-K decoding separately and at equal wall-clock time, including the domains in which ELDOR ties or loses.

2 Dismantling and Two Ways to Value a Removal

We introduce the notation, the objective and the two ways of valuing a removal; detailed notation, related work and the full protocol are in Appendices A, B and K, respectively.

Network Dismantling. Let $G = (V, E)$ be a simple undirected graph with $n = |V|$ nodes and $m = |E|$ edges; $G[S]$ is the subgraph induced by $S \subseteq V$ and $C_{\max}(S)$ its largest connected component. A *removal trajectory* of budget $B \leq n$ is a sequence $\pi = (v_1, \ldots, v_B)$ of distinct nodes, which induces nested surviving sets $S_t = V \setminus \{v_1, \ldots, v_t\}$ with $S_0 = V$; $b_t = B - t$ is the remaining budget and $c(S) = |C_{\max}(S)|/n$ the normalized giant component. A dismantler is trained on a few graph families and sizes, and must then transfer to graphs of unseen structure and size.

Dismantling Objectives. Following the standard protocol (Grassia et al., 2021; Artime et al., 2024), nodes are removed until the giant component falls below ρn ($\rho = 0.1$). With the collapse time

$B_\rho(\pi; G) = \min\{t : |C_{\max}(S_t)| \le \rho n\}$ and the fixed-budget residual $\Gamma(\pi; G) = c(S_B)$, dismantling performance is measured by the area under the dismantling curve up to collapse,

$$\mathcal{R}(\pi; G) = \frac{1}{n} \sum_{t=0}^{B_\rho - 1} c(S_t), \tag{1}$$

reported $\times 100$ with unit weight per pre-collapse state. We use no Simpson weighting, so the reported metric differs from the optimized objective below only by a constant (Appendix K.1). All metrics, including the pairwise connectivity $\mathcal{P}$ (Appendix A), are lower-is-better. Γ depends only on the set of removed nodes, whereas $\mathcal{R}$ also depends on their order (Figure 1b). With $c_\rho(S) = c(S)\mathbb{1}[|C_{\max}(S)| > \rho n]$, $n\mathcal{R} = c(V) + \sum_{t=1}^{B} c_\rho(S_t)$ for every $B \ge B_\rho$, where $c(V)$ is fixed by G ($c(V) = 1$ for a connected graph), so minimizing $\mathcal{R}$ minimizes the *path cost* $\Phi(\pi) = \sum_{t=1}^{B} c_\rho(S_t)$; we write c for c_ρ below. We assume uniform removal costs on a fully observed single-layer graph; costs, defenses, layers and hyperedges change the immediate term (Appendix C.1).

Two Ways to Value a Removal. A dismantler selects the next removal from logits $\ell(v; S, b)$, and end-to-end methods learn the whole logit (Fan et al., 2020; Tian et al., 2026). The same kernels can, however, be written differently: any strictly positive kernel equals $q \propto q_0\, e^{-\beta c(S\setminus\{v\})} e^{r}$ with $r = \ell - \log q_0 + \beta c(S \setminus \{v\})$: the two specifications represent exactly the same kernels and differ only in what is learned. In the second, the immediate term is computed rather than inferred, and the learned term has an explicit target, $\log h_{b-1}$ (Section 3.2).

Structure-only Input and Transfer. We call a dismantler *structure-only* if its only inputs are G, B and optionally ρ, together with quantities computed from them. Unlike the constant initialization of MIND (Tian et al., 2026), this admits exact structural inputs such as degree (Section 3.2). A dismantler is *permutation-equivariant* if relabeling the graph relabels its trajectory distribution. We test transfer in the standard zero-shot sense: trained once on synthetic graphs, a method is evaluated on unseen test networks without retraining or per-dataset tuning.

3 ELDOR: Doob-Controlled Removal Trajectories

ELDOR models the removal process itself rather than its outcome. The state is the set of surviving nodes and every transition removes one of them; a reference process that knows only the current topology proposes removals, and a budget-conditioned potential reweights them by how much each removal is expected to fragment the graph in the remaining steps. The immediate effect of every removal is computed exactly, and only the future part is estimated by a structure-only network that operates on the residual graph and therefore handles graphs of any size (Figure 2).

3.1 Absorbing Removal Process

Reference Kernel. Solution-denoising models generate a complete solution vector and recover an ordering only afterwards, so their intermediate states need not correspond to any sequence of interventions. We therefore define the process directly on the sets of surviving nodes. A *removal kernel* assigns to each state (S, b) with $1 \le b \le |S|$ a distribution over the children of S, i.e., over the sets obtained by removing one surviving node:

$$q(S' \mid S, b, G) = 0 \text{ unless } S' \subset S,\ |S \setminus S'| = 1; \quad \text{we write } q(v \mid S, b) := q(S \setminus \{v\} \mid S, b). \tag{2}$$

The *reference kernel* is the simplest such rule that depends only on the current topology:

$$q_0(v \mid S) = \frac{w(v; S)}{\sum_{u \in S} w(u; S)}, \qquad w(v; S) = \big(1 + \deg_{G[S]}(v)\big)^{\alpha},\ \alpha \ge 0, \tag{3}$$

where $\alpha = 0$ gives uniform removal and $\alpha > 0$ a soft preference for high-degree nodes. The reference kernel is deliberately uninformed. Its role is to guarantee that every surviving node remains selectable, so that the controlled process can only reweight valid removals. Each survival indicator thus follows an absorbing two-state chain, as in the forward process of absorbing discrete diffusion (Austin et al., 2021), but the chains are coupled through $G[S_t]$ and nothing is denoised: ELDOR learns a potential that controls this forward chain (Appendix D). States with $b = 0$ are terminal, and multi-node removal per evaluation composes single-node steps from a frozen kernel (Section 3.3).

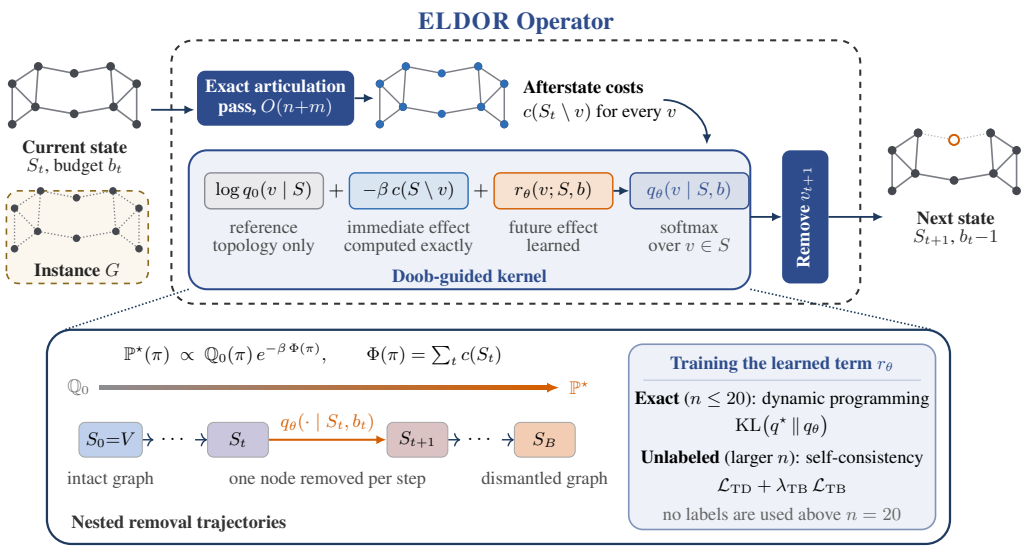

Figure 2: **Overview of ELDOR. Top:** one operator call. An exact $O(n+m)$ articulation pass gives the afterstate cost of every candidate, and the Doob-guided kernel adds the topology-only reference term (grey), this exact term (blue) and the learned future term r_θ (orange) before one node is removed. In the toy graph no single removal disconnects the graph, so the exact term is flat and only r_θ can single out the two connectors. **Bottom:** with the exact potential, the kernel samples nested removal trajectories from the reference law $\mathbb{Q}_0$ tilted toward early fragmentation ($\mathbb{P}^\star$); r_θ is trained against exact dynamic-programming targets on small graphs and by self-consistency on larger unlabeled ones.

Critical Structural Clock. A uniform time index t/B says little about the progress of dismantling: most removals barely change the giant component, while a few cause it to collapse. We therefore measure progress by the structure of the residual graph itself, using an exact primitive.

Lemma 1 (All-children giant components in linear time)**.** *Given $G[S]$, the values $|C_{\max}(S \setminus \{v\})|$ for all $v \in S$, and the sizes of the pieces into which each removal splits its component, can be computed in $O(|S| + |E(G[S])|)$ time.*

The proof follows from depth-first search with low-points (Tarjan, 1972) (Appendix F). Using Lemma 1, we define the *articulation sensitivity* $\xi(v; S) = (|C_{\max}(S)| - |C_{\max}(S \setminus \{v\})|)/|C_{\max}(S)| \in [0, 1]$ of each node, and summarize the state by the critical structural clock:

$$\tau(S) = 1 - \frac{|C_{\max}(S)|}{|C_{\max}(V)|}, \qquad \tilde{\chi}(S) = \frac{\sum_{C \neq C_{\max}} |C|^2}{|C_{\max}(S)| \sum_{C \neq C_{\max}} |C|}, \qquad \Delta^*(S) = \max_{v \in S} \xi(v; S), \quad (4)$$

with $\tilde{\chi}(S) = 0$ if $G[S]$ is connected. Here τ is the *structural time*, which is nondecreasing along every trajectory (Lemma 6); $\tilde{\chi}$ is the percolation susceptibility (Stauffer & Aharony, 1994) relative to the giant component; and Δ^* is the largest relative collapse that a single removal can achieve. Their combination $\varrho(S) = \max\{\tilde{\chi}(S), \Delta^*(S)\} \in [0, 1]$, which we call the *criticality density*, is large exactly when the graph is close to fragmenting.

The idea behind the clock is to spend modeling capacity where decisions matter: i) the network is conditioned on $\gamma(S, b) = (\tau, \varrho, \tilde{\chi}, \Delta^*, b/|S|)$ instead of t/B; ii) the number of nodes removed per network evaluation is $k(S) = \min\{b, \max\{1, \lfloor k_{\max}(1 - \varrho(S)) \rfloor\}\}$, so evaluations concentrate near fragmentation; and iii) training states are weighted by $\epsilon_w + \varrho(S)$. All three are computed from the state that the sampler has actually reached, so the clock cannot leak an optimal solution.

3.2 Doob-Guided Removal Dynamics

Desirability Potential. To bias the reference process toward early collapse, each state needs a score that reflects not only its current giant component but also what the remaining budget can still

achieve. We fix an inverse temperature $\beta > 0$, a running weight $g(S) = e^{-\beta c(S)}$ that penalizes large residual giant components, and a terminal weight $\phi(S) > 0$; the evaluated parameterization instantiates $\phi \equiv 1$. The *desirability* of a state with budget b is then given by the backward recursion

$$h_0(S) = \phi(S), \qquad h_b(S) = \sum_{v \in S} q_0(v \mid S)\, g(S \setminus \{v\})\, h_{b-1}(S \setminus \{v\}), \quad 1 \le b \le |S|. \tag{5}$$

Unrolling Eq. (5) gives $h_b(S) = \mathbb{E}_{q_0}[\prod_{j=1}^{b} g(S_{t+j})\phi(S_{t+b}) \mid S_t = S]$, the expected exponentiated negative cost of the remaining trajectory under the reference process, so $V_b = -\beta^{-1} \log h_b$ is the soft cost-to-go relative to q_0 (Proposition 5). Intuitively, a state is desirable when many of its continuations fragment the graph early; the budget enters through b, and since the successor $S \setminus \{v\}$ is determined by the action, $V_{b-1}(S \setminus \{v\})$ is an afterstate value (Sutton & Barto, 2018).

Exact and Learned Control. Given the desirability, the removal kernel that follows the reference process but favors desirable successors is the discrete Doob transform of the g-weighted reference kernel:

$$q^\star(v \mid S, b) = \frac{q_0(v \mid S)\, g(S \setminus \{v\})\, h_{b-1}(S \setminus \{v\})}{h_b(S)}. \tag{6}$$

By Eq. (5), the denominator equals the sum of the numerators, so $q^\star$ is normalized, and it vanishes wherever q_0 does, so removed nodes can never be selected again; with $g \equiv 1$ it reduces to the classical h-transform. Running this kernel from V for B steps produces trajectories with the Feynman–Kac path law (Theorem 2)

$$\mathbb{P}^\star(\pi \mid G, B) = \frac{1}{h_B(V)} \prod_{t=1}^{B} q_0(v_t \mid S_{t-1})\, e^{-\beta c(S_t)}\, \phi(S_B) = \frac{\mathbb{Q}_0(\pi)\, e^{-\beta \Phi(\pi)} \phi(S_B)}{h_B(V)}, \tag{7}$$

i.e., the reference path law reweighted by the exponentiated cost of the whole trajectory (Figure 2, bottom).

However, Eq. (6) is exact only if h is the true desirability, which can be computed for small graphs alone. ELDOR therefore replaces the unknown future term $\log h_{b-1}(S \setminus \{v\})$ by a learned estimate $r_\theta(v; S, b)$ and obtains the Doob-guided kernel:

$$q_\theta(v \mid S, b) = \frac{\exp \ell_\theta(v; S, b)}{\sum_{u \in S} \exp \ell_\theta(u; S, b)}, \qquad \ell_\theta(v; S, b) = \log q_0(v \mid S) - \beta c(S \setminus \{v\}) + r_\theta(v; S, b), \tag{8}$$

with implied set-level log-potential $u_\theta(S, b) = \mathrm{LSE}_{v \in S}\, \ell_\theta(v; S, b)$ and $u_\theta(\cdot, 0) = \log \phi$. Equation (8) separates what can be computed from what must be learned: the reference term guarantees valid removals for every θ, the immediate term is computed exactly for all v by Lemma 1, and the network is responsible only for the future consequences of a removal under the remaining budget. For every θ, q_θ keeps the support of Eq. (2), and it coincides with $q^\star$ if and only if $r_\theta = \log h_{b-1}$ up to a node-independent shift (Proposition 9); $r_\theta \equiv 0$ gives the myopic exact-greedy kernel, evaluated as an ablation (Table 3).

Equivariant Parameterization. To keep the learned term independent of node identities and graph size, the network receives only the residual graph $G[S]$, the budget b and the clock $\gamma(S, b)$. Node inputs are recomputed from $G[S]$ at every step—log-degree and its deviation from the neighborhood mean, ξ, the log-number of pieces a removal creates, giant-component membership and relative component size—and none depends on node identities or directly on n. A residual message-passing network with mean/max aggregation and FiLM conditioning on γ (Perez et al., 2018) maps them to node states $\mathbf{e}_v^{(L)}$ and a graph readout $\bar{\mathbf{e}}$ (Appendix I). Degree thus enters explicitly as an input, and α, which only biases the reference kernel, has little effect (**??**); whether the learned term relies on degree shortcuts is tested separately, on held-out degree-trap graphs (Table 18).

We further constrain the potential head by the structure of the problem. Since c is nonincreasing along trajectories (Lemma 6), the soft cost-to-go of a successor satisfies $0 \le V_{b-1}(S \setminus \{v\}) \le (b-1)\, c(S \setminus \{v\})$ (Proposition 5), and we parameterize the future term within these bounds:

$$r_\theta(v; S, b) = -\beta\, (b-1)\, c(S \setminus \{v\})\, \sigma\big(z_\theta(v; S, b)\big), \qquad z_\theta(v) = \mathrm{MLP}\big([\mathbf{e}_v^{(L)}, \bar{\mathbf{e}}, \gamma]\big). \tag{9}$$

The first motivation for this form is correctness at the boundary: $r_\theta = 0$ when $b = 1$, matching $h_0 = \phi \equiv 1$. The second motivation is scale: the head outputs a fraction of the largest possible

future cost, which remains meaningful as n and b change; we set $\beta = \bar{\beta} n$ and evaluate the head in double precision (Appendix I). The backbone is 1-WL-bounded (Xu et al., 2019); the exact inputs supply biconnectivity information it cannot compute on its own (Zhang et al., 2023).

3.3 Training and Decoding

Exact targets exist only on small graphs, and imitating heuristics would import their biases (Appendix J). We therefore combine two sources of supervision: exact dynamic programming on small graphs, and self-consistency of the potential on larger unlabeled graphs.

Exact Regime ($n \le n_{\mathrm{ex}} = 20$). On small training graphs, $\log h_b(S)$ is computed for all reachable states by the dynamic program of Eq. (5) in $O(2^n(n+m))$ time (Appendix G), and over the visited states $\mathcal{D}_{\mathrm{ex}}$ the network matches both the exact kernel and the exact potential:

$$\mathcal{L}_{\mathrm{ex}}(\theta) = \mathbb{E}_{(S,b)\sim\mathcal{D}_{\mathrm{ex}}}\Big[D_{\mathrm{KL}}\big(q^\star(\cdot \mid S,b) \,\|\, q_\theta(\cdot \mid S,b)\big) + \lambda_V \big(u_\theta(S,b) - \log h_b(S)\big)^2 / s_{\mathrm{ex}}(S,b)^2 \Big], \tag{10}$$

with $s_{\mathrm{ex}}(S,b) = \beta \max\{b\, c(S), n^{-1}\}$.

Collapsed states ($c(S) = 0$) are terminal ($h_b = 1$, $q^\star = q_0$): they are excluded from $\mathcal{D}_{\mathrm{ex}}$, rollouts stop there and targets are $u = 0$ (Appendix J.1).

Unlabeled Regime. For larger graphs G with budgets $B \sim \mathrm{Unif}\{\lceil f_{\min} n \rceil, \dots, \lceil f_{\max} n \rceil\}$ ($f_{\min} = 0.1$, $f_{\max} = 0.5$), we sample trajectories from a behavior kernel $\mu = (1-\epsilon_\mu)\, q_\theta^{(T)} + \epsilon_\mu\, q_0$, where $q_\theta^{(T)}$ is q_θ at sampling temperature T (Section 3.3), so that every removal remains possible. Since the true potential satisfies a one-step consistency condition, we penalize its violation through the residual

$$\delta_t(\theta) = r_\theta(v_{t+1}; S_t, b_t) - \mathrm{sg}\left[u_{\bar{\theta}}(S_{t+1}, b_t - 1)\right] \quad (\bar{\theta} \leftarrow \kappa\bar{\theta} + (1-\kappa)\theta,\ u(\cdot,0) = 0), \tag{11}$$

which gives the per-step and trajectory-level losses

$$\mathcal{L}_{\mathrm{TD}} = \mathbb{E}\Big[\tfrac{\epsilon_w + \varrho(S_t)}{Z_w} \big(\delta_t / s_t\big)^2\Big], \qquad \mathcal{L}_{\mathrm{TB}} = \mathbb{E}_{\pi\sim\mu}\Big[\Big(\textstyle\sum_{t=0}^{B-1} \delta_t^\circ(\theta)\Big)^2 \Big/ (\beta B)^2\Big], \tag{12}$$

where $s_t = \beta \max\{(b_t - 1)c(S_{t+1}), n^{-1}\}$ and Z_w are normalizers, and δ_t° denotes Eq. (11) with $\bar{\theta} = \theta$ and no stop-gradient. $\mathcal{L}_{\mathrm{TB}}$ is a trajectory-balance loss (Malkin et al., 2022; Nüsken & Richter, 2021) whose zero set is $\{\mathbb{P}_\theta = \mathbb{P}^\star,\ u_\theta(V,B) = \log h_B(V)\}$ (Proposition 11), and $\mathcal{L}_{\mathrm{TD}}$ is its lower-variance per-step counterpart (Nachum et al., 2017); the clock weights $\epsilon_w + \varrho$ do not change the exact minimizers (Proposition 9) but concentrate the learning signal near fragmentation.

Optimization. The overall objective at training step i is $\mathcal{L} = \lambda_{\mathrm{ex}}(i)\mathcal{L}_{\mathrm{ex}} + \mathcal{L}_{\mathrm{TD}} + \lambda_{\mathrm{TB}}\mathcal{L}_{\mathrm{TB}}$. Training begins on small graphs, where exact targets anchor the potential, and gradually moves to larger unlabeled graphs, where only consistency is enforced: λ_{ex} decays to a floor $\lambda_{\mathrm{ex}}^{\min} > 0$, the maximum training size grows from n_{ex} to $\max \mathcal{N}_{\mathrm{tr}}$, and $\bar{\beta}$ is annealed upward (schedules in Table 8; procedure in Algorithm 3). No labels are used above n_{ex}.

Decoding. At inference time, ELDOR repeatedly evaluates Eq. (8) on the residual graph and removes the selected nodes (Algorithm 4), optionally stopping once $|C_{\max}| \le \rho n$. Because $\beta = \bar{\beta} n$, sampling from $\mathrm{softmax}(\ell_\theta / (\bar{\beta} T))$ makes a one-node difference in afterstate cost one nat at $T = 1$ on every graph size. We distinguish two decoding modes and two search procedures. *Sequential* decoding ($k_{\max} = 1$), greedy with random tie-breaking, which preserves equivariance, or sampled, follows the kernel of Section 4; *batched* decoding draws $k(S)$ nodes from frozen logits by Gumbel-top-k (Kool et al., 2019b) and removes them in the drawn order, an acceleration that is no longer a sample of that kernel (Table 19). Best-of-K selection and local refinement are search on top of either mode. Greedy decoding measures amortized quality, best-of-K search-enhanced quality. Candidates are scored exactly by reverse union–find reinsertion (Newman & Ziff, 2000), and refinement keeps a local move only if it lowers $\mathcal{R}$.

Inference costs $O\big(K T_{\mathrm{ev}}(L(n+m)d + Lnd^2)\big)$ time with $T_{\mathrm{ev}} \in [\lceil B/k_{\max} \rceil, B]$ network evaluations (Proposition 12), i.e. $\Theta(n(n+m))$ for $B = \Theta(n)$ without batching.

4 What the Theory Guarantees

It is useful to keep three objects apart: (A) the exact process driven by $q^\star$, (B) the learned kernel q_θ as a function of its potential, and (C) training and decoding. The statements below concern (A) and (B); what is and is not established at each level is collected in Appendix C.2, and the proofs are in Appendices E and F.

Validity and Equivariance. For *every* θ, any kernel satisfying Eq. (2) yields strictly nested trajectories of exactly B distinct removals that never reselect a node (Proposition 3), and with the equivariant r_θ of Section 3.2 the trajectory law satisfies $\mathbb{P}_\theta(\sigma\pi \mid \sigma G, B) = \mathbb{P}_\theta(\pi \mid G, B)$ under all decoding modes (Proposition 4); both hold by construction.

Theorem 2 (Exact Doob/Feynman–Kac control)**.** *Let $g > 0$, $\phi > 0$, $B \le n$, and h be defined by Eq. (5). Then (i) $q^\star(\cdot \mid S, b)$ in Eq. (6) is a probability distribution for every S and $1 \le b \le |S|$; (ii) $q^\star(v \mid S, b) > 0$ if and only if $v \in S$, i.e. $q^\star$ and q_0 have the same support; (iii) the Markov chain started at V and driven by $q^\star(\cdot \mid S_t, B - t)$ for B steps has trajectory law $\mathbb{P}^\star$ in Eq. (7).*

With the exact potential, the process therefore samples a path measure that favors early fragmentation; for $\phi \equiv 1$, $V_b = -\beta^{-1} \log h_b$ lies within $\beta^{-1} b(\log n + \alpha \log(1 + \Delta_G))$ of the hard cost-to-go, so $\mathbb{P}^\star$ concentrates on near-optimal trajectories as β grows (Proposition 5).

From Exact to Learned Control. ELDOR learns the potential that Theorem 2 assumes, and the learning results identify the idealized fixed point of this learning: with a full-support behavior kernel and $\bar\theta = \theta$, a zero consistency residual identifies $q^\star$ (Propositions 9 and 11), and $|r_\theta - \log h_{b-1}| \le \epsilon$ on all reachable states gives $D_{\mathrm{KL}}(\mathbb{P}_\theta \| \mathbb{P}^\star) \le 2\epsilon B$ (Proposition 8). They do not assert that the bounded head of Eq. (9) reaches a zero residual on large graphs, or that lagged-target training converges to it; the uniform premise can be checked only where $\log h$ is exact, which Table 17 does up to $n = 20$; on the test sets we report consistency diagnostics, which support approximate consistency but do not establish the premise. The path-law result holds for sequential decoding ($k_{\max} = 1$) and any $\phi > 0$; the bounds and the head use $\phi \equiv 1$.

A Degree Trap. On the trap family of Algorithm 2, degree and adaptive-degree removal take all k hubs first while $|C_{\max}(S_t)| = n - t$, the worst attainable value, whereas removing the $r(k-1)$ connectors leaves components of size $s+1$ (Proposition 7); this covers only the first k removals and does not defeat betweenness (Appendix H).

5 Experiments

5.1 Protocol

We follow the evaluation protocol of Tian et al. (2026): learned methods are trained once on synthetic graphs and evaluated zero-shot, without retraining or per-dataset tuning, on two test sets held out from training: the 48 real networks of the benchmark (9 biological, 14 information, 16 social and 9 technological networks with 128 to 1,402,673 nodes; Table 11) and the synthetic test protocol (Erdős–Rényi, configuration-model and SBM graphs with 10^3 to 10^5 nodes; Appendix L). For ELDOR, the configuration-model graphs are an unseen family and every size above 10^3 exceeds its largest training graph (validation graphs, used only for checkpoint selection, reach 1,500 nodes).

Training Data and Baselines. ELDOR is trained on seven synthetic families (ER, BA, WS, SBM, RGG, planted-partition, fragmentation; Table 6). We compare with nine published methods, whose reported results we quote: adaptive degree (AD), betweenness (BC), PageRank (PR), min-sum (MS), explosive immunization (EI) (Clusella et al., 2016), GND (Ren et al., 2019), FINDER (Fan et al., 2020), GDM (Grassia et al., 2021) and MIND (Tian et al., 2026). We further run six methods under the same protocol: CoreHD (Zdeborová et al., 2016), reverse dismantling (RD) in its dynamic variant (Duan et al., 2026), the recent LGD-NA and its common-neighbor rule CND (Adler et al., 2026), NIRM trained on tiny synthetic networks with its published protocol (Zhang & Wang, 2022), and Structural-AR (ELDOR's encoder with direct node scoring, which isolates the Doob potential).

Metrics and Implementation Details. We report the dismantling AUC of Eq. (1), normalized per network (MIND baseline $= 100$), with runtime and memory. Quoted values use Simpson's rule; recomputing them under Eq. (1) changes none by more than 0.4 and no average rank (Appendix K.1).

Table 1: Zero-shot dismantling AUC on the 48 real test networks ($\rho = 0.1$), normalized per network so that MIND $= 100$ (↓): mean per domain (number of networks in parentheses) and overall, and average rank over 17 ranked entries (greedy and best-of-16 ELDOR ranked separately). Src.: Q published values (Tian et al., 2026) (Simpson's rule, `n/a` excluded; Appendix K.1); R run under the same protocol, learned methods averaged over 5 trained models (SDs for ELDOR in Table 16). Δ: mean paired difference to LGD-NA over networks, negative is better (intervals and tests in Table 22). The batched row approximates greedy ELDOR and is not ranked; every network is in Table 14. Shading marks the best (bold) and second-best value in each column.

Method	Src.	Bio (9)	Info (14)	Social (16)	Tech (9)	All (48)	Δ vs. LGD-NA	Avg. rank↓
Adaptive degree (AD)	Q	110.4	122.7	110.3	141.8	119.9	+26.0	12.6
Betweenness (BC)	Q	137.8	193.3	129.6	225.8	167.8	+73.9	15.7
PageRank (PR)	Q	125.6	133.7	116.8	149.4	129.5	+35.6	14.1
Min-sum (MS)	Q	129.5	138.3	132.7	180.5	142.8	+49.0	15.3
Explosive immunization (EI)	Q	121.7	109.0	122.2	**67.4**	108.0	+14.1	10.8
GND	Q	118.6	118.0	112.7	80.3	109.1	+15.2	11.1
FINDER	Q	106.9	118.0	107.1	132.4	115.0	+21.1	10.4
GDM	Q	105.6	105.0	104.7	100.8	104.2	+10.4	8.8
MIND	Q	100.0	100.0	100.0	100.0	100.0	+6.1	6.2
NIRM	R	102.7	103.3	103.7	110.2	104.6	+10.8	9.0
Structural-AR	R	104.9	104.4	104.7	112.2	106.0	+12.2	9.8
CoreHD	R	108.2	118.8	108.2	130.8	115.5	+21.7	10.6
CND	R	99.3	99.4	100.4	84.0	96.8	+3.0	5.8
Reverse dismantling (RD)	R	97.7	97.7	97.9	76.0	93.7	−0.2	3.9
LGD-NA	R	97.0	96.6	97.2	80.5	93.9	–	3.5
ELDOR greedy	R	95.4	95.0	95.7	83.8	93.2	−0.6	3.3
ELDOR greedy, batched (approx.)	R	95.8	95.3	96.1	84.2	93.6	−0.3	–
ELDOR best-of-16	R	**92.4**	**92.0**	**92.7**	80.8	**90.2**	−3.7	1.8

ELDOR is run with $B = \lceil 0.3n \rceil$; if $|C_{\max}| > \rho n$ at the end, the trajectory is recomputed with B raised by $\lceil 0.1n \rceil$ (budgets above $0.5n$ lie outside the training range and are marked). We separate policy quality (sequential greedy, $k_{\max} = 1$, the kernel of Section 4), its batched engineering approximation ($k_{\max} = 16$), search (best-of-16) and quality at equal wall-clock time. Uncertainty is the SD across 5 trained models, paired tests use networks as units (Table 22), and checkpoints are selected on synthetic validation cells only (Appendix K). Code is provided in the supplementary material.

5.2 Results on Real Networks

Policy Quality. Sequential greedy ELDOR lowers the average AUC from 100 (MIND baseline) to 93.2 (Table 1), improves on MIND on 42 networks and on NIRM by 11 points, and is on par with the two strongest training-free baselines overall, LGD-NA (-0.6) and reverse dismantling (-0.5; neither difference is significant). Against LGD-NA it is significantly better on the 39 biological, information and social networks (-1.6, $p < 10^{-4}$; also within each domain) and significantly worse on the nine technological ones, where explosive immunization (67.4), reverse dismantling (76.0) and LGD-NA (80.5) remain ahead. Batching, an engineering approximation, costs 0.4 points.

Search and Fixed Compute. Best-of-16 reaches 90.2 (rank 1.8), the best average of all methods and significantly better than LGD-NA (-3.7), reverse dismantling (-3.5) and LGD-NA restarted for the same time (-2.0, all $p < 10^{-4}$; Table 22); per-network results are in Table 14.

5.3 Synthetic Test Protocol

On the synthetic protocol (Table 2), greedy ELDOR reaches 94.9 and best-of-16 91.7, against 100 for MIND and 98.0 for LGD-NA. The greedy margin over LGD-NA shrinks from 4.0 points on cells that share family and size with training to 3.1 at unseen sizes and 2.4 on the unseen configuration-model family, but stays positive in every group.

5.4 Model Analyses

Contribution of Each Component. In Table 3, replacing the learned Doob potential by direct scoring (Structural-AR) costs the most, followed by removing absorption and the exact term; the

Table 2: Synthetic test protocol, dismantling AUC ($\rho = 0.1$) normalized so that MIND $= 100$ (↓). Columns average all 90 graphs and three groups: family and size seen in training ($n = 10^3$, ER and SBM), seen families at unseen sizes ($n \geq 10^4$), and the unseen configuration-model (CM) family. All methods and cells are in Table 12. Shading marks the best (bold) and second-best value in each column.

Method	Overall	Seen family, seen size	Seen family, unseen size	Unseen family (CM)
MIND	100.0	100.0	100.0	100.0
NIRM	101.5	100.9	101.6	101.7
Structural-AR	102.6	102.5	102.5	102.7
CoreHD	103.6	104.2	103.4	103.3
CND	100.8	100.8	100.3	101.4
RD	99.1	99.1	98.7	99.6
LGD-NA	98.0	97.8	97.6	98.6
ELDOR greedy	94.9	93.8	94.5	96.2
ELDOR best-of-16	**91.7**	**91.3**	**91.2**	**92.5**

Table 3: Ablation of ELDOR (sequential greedy, $k_{\max}$=1) on the test sets, dismantling AUC normalized so that MIND $= 100$ (↓), for the real networks by domain and the synthetic protocol. Shading marks the best value (bold) and the best ablated variant in each column.

	Real networks					
Variant	Bio	Info	Social	Tech	All (48)	Synthetic protocol
ELDOR (full)	**95.4**	**95.0**	**95.7**	**83.8**	**93.2**	**94.9**
− Doob guidance: Structural-AR head	104.9	104.4	104.7	112.2	106.0	102.6
Myopic exact greedy ($r_\theta \equiv 0$)	103.6	103.2	103.9	91.0	101.2	103.0
− exact immediate term	104.8	104.4	105.1	92.1	102.4	104.3
− absorption (reversible bit-flip chain)	106.8	106.3	107.1	93.8	104.3	106.2
− budget conditioning	99.2	98.8	99.5	87.1	96.9	98.7
uniform t/B clock	100.8	100.4	101.1	88.6	98.5	100.3
− exact small-graph supervision	100.4	100.0	100.7	88.2	98.1	99.9
− consistency training	102.5	102.0	102.8	90.0	100.1	101.9

myopic kernel is also clearly worse, so both parts of the split matter, and the remaining components give smaller, consistent gains. On controlled holdouts (Table 18), the greedy margin over LGD-NA grows with separator width, from 3.5 points at $w = 3$ to 4.7 at $w = 4$, all methods tie within 1 point on near-symmetric candidates, and dynamic betweenness remains best on non-articulation cuts.

Potential Error and Consistency. The uniform premise of Proposition 8 can be checked only where $\log h$ is exact: up to $n = 20$, $\hat{\epsilon}_{\max} \leq 0.34$ nats and the path divergence is about $10\times$ below the bound (Table 17). On the test sets we report only consistency diagnostics (residual ≤ 0.052, rollout agreement $\rho_s \geq 0.86$), which indicate approximate consistency, not the uniform bound.

Decoding and Compute. At $n = 10^3$, sequential greedy takes 7.4 s per graph ($15\times$ LGD-NA), batched greedy 0.93 s and best-of-16 14.7 s (Table 20); at the batched time budget restarted LGD-NA reaches 93.7 against 93.6, and batching changes any network by at most 1.8 points (Table 19).

6 Conclusion

Separating what can be computed from what must be learned helps dismantlers transfer: a Doob transform reweights a topology-only reference process, the immediate term is exact, and a structure-only network learns only the budget-dependent future. Greedy ELDOR improves on learned dismantlers and is on par with LGD-NA and reverse dismantling; best-of-16 is significantly better than both, also against LGD-NA restarted for equal time. The method assumes fully observed, unweighted, single-layer graphs with uniform costs, its guarantee is checked exactly only up to $n = 20$, and separator-based heuristics stay ahead on technological networks (Appendices C.1 and M.1).

USE OF LARGE LANGUAGE MODELS

Large language models were used only for copyediting and routine coding support; they did not generate ideas, results or analyses, and the authors verified all text and code and take full responsibility for the paper.

ETHICS STATEMENT

Dismantling algorithms support robustness assessment, immunization and redundancy planning, but the same outputs could in principle guide the deliberate disruption of infrastructure or social networks. We adopt two safeguards: we evaluate only on publicly available, previously published networks, and we do not release removal sequences for the real infrastructure networks used in this work; only aggregate metrics are reported. The released code and checkpoints can regenerate such sequences for any input network, so this safeguard only avoids publishing ready-made removal orders; comparable heuristics (adaptive degree, betweenness) are already widely available. The study involves no human subjects, no personally identifiable information and no crowdsourcing, and all datasets are used under the licenses of their originating repositories.

REPRODUCIBILITY STATEMENT

The complete experimental protocol—generator families and parameter ranges, splits, budgets, baseline tuning and evaluation metrics—is given in Appendix K and summarized in Tables 6 to 9. Training and inference are specified in pseudocode (Algorithms 1 to 4), architecture and numerical details in Appendix I, and full proofs of all formal statements in Appendix F. Every learned method is trained with five independent seeds and evaluated with fixed decoding seeds; uncertainty and significance testing follow Appendix K and Table 22, and compute and hardware are reported in Table 10. Code, data generators and trained checkpoints are provided in the supplementary material.

REFERENCES

Thomas Adler, Marco Grassia, Ziheng Liao, Giuseppe Mangioni, and Carlo V. Cannistraci. Latent geometry-driven network automata for complex network dismantling. In *International Conference on Learning Representations*, 2026.

Réka Albert, Hawoong Jeong, and Albert-László Barabási. Error and attack tolerance of complex networks. *Nature*, 406:378–382, 2000. doi: 10.1038/35019019.

Oriol Artime, Marco Grassia, Manlio De Domenico, James P. Gleeson, Hernán A. Makse, Giuseppe Mangioni, Matjaž Perc, and Filippo Radicchi. Robustness and resilience of complex networks. *Nature Reviews Physics*, 6:114–131, 2024. doi: 10.1038/s42254-023-00676-y.

Ashwin Arulselvan, Clayton W. Commander, Lily Elefteriadou, and Panos M. Pardalos. Detecting critical nodes in sparse graphs. *Computers & Operations Research*, 36(7):2193–2200, 2009. doi: 10.1016/j.cor.2008.08.016.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, 2021.

Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286: 509–512, 1999. doi: 10.1126/science.286.5439.509.

Ulrik Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25(2):163–177, 2001. doi: 10.1080/0022250X.2001.9990249.

Alfredo Braunstein, Luca Dall'Asta, Guilhem Semerjian, and Lenka Zdeborová. Network dismantling. *Proceedings of the National Academy of Sciences*, 113:12368–12373, 2016. doi: 10.1073/pnas.1605083113.

Pau Clusella, Peter Grassberger, Francisco J. Pérez-Reche, and Antonio Politi. Immunization and targeted destruction of networks using explosive percolation. *Physical Review Letters*, 117: 208301, 2016. doi: 10.1103/PhysRevLett.117.208301.

Reuven Cohen, Keren Erez, Daniel ben Avraham, and Shlomo Havlin. Breakdown of the internet under intentional attack. *Physical Review Letters*, 86:3682–3685, 2001. doi: 10.1103/PhysRevLett.86.3682.

Pierre Del Moral. *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications*. Probability and its Applications. Springer, 2004. doi: 10.1007/978-1-4684-9393-1.

Joseph L. Doob. Conditional Brownian motion and the boundary limits of harmonic functions. *Bulletin de la Société Mathématique de France*, 85:431–458, 1957. doi: 10.24033/bsmf.1494.

Jinyu Duan, Sijin Wang, Fan Zhang, Xiang Zhao, Wenjie Zhang, and Zhihong Tian. Network dismantling via reverse dismantling: Static and dynamic algorithms. In *Proceedings of the ACM Web Conference*, 2026.

Paul Erdős and Alfréd Rényi. On random graphs I. *Publicationes Mathematicae Debrecen*, 6: 290–297, 1959.

Changjun Fan, Li Zeng, Yizhou Sun, and Yang-Yu Liu. Finding key players in complex networks through deep reinforcement learning. *Nature Machine Intelligence*, 2:317–324, 2020. doi: 10.1038/s42256-020-0177-2.

Linton C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977. doi: 10.2307/3033543.

Marco Grassia, Manlio De Domenico, and Giuseppe Mangioni. Machine learning dismantling and early-warning signals of disintegration in complex systems. *Nature Communications*, 12:5190, 2021. doi: 10.1038/s41467-021-25485-8.

Weiwei Gu, Chen Yang, Lei Li, Jinqiang Hou, and Filippo Radicchi. Deep-learning-aided dismantling of interdependent networks. *Nature Machine Intelligence*, 7, 2025. doi: 10.1038/s42256-025-01070-2.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning*, 2017.

Paul W. Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social Networks*, 5(2):109–137, 1983. doi: 10.1016/0378-8733(83)90021-7.

Li Hong, Mengqiao Xu, Yu Liu, Xuemei Zhang, and Changjun Fan. A deep learning dismantling approach for understanding the structural vulnerability of complex networks. *Chaos, Solitons & Fractals*, 194:116148, 2025. doi: 10.1016/j.chaos.2025.116148.

Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019a.

Wouter Kool, Herke van Hoof, and Max Welling. Stochastic beams and where to find them: The Gumbel-top-k trick for sampling sequences without replacement. In *International Conference on Machine Learning*, 2019b.

Qingxia Liu and Bang Wang. Neural extraction of multiscale essential structure for network dismantling. *Neural Networks*, 154:99–108, 2022.

Andreas Loukas. What graph neural networks cannot learn: depth vs width. In *International Conference on Learning Representations*, 2020.

Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in GFlowNets. In *Advances in Neural Information Processing Systems*, 2022.

Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. In *International Conference on Learning Representations*, 2019.

Flaviano Morone and Hernán A. Makse. Influence maximization in complex networks through optimal percolation. *Nature*, 524:65–68, 2015. doi: 10.1038/nature14604.

Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. In *Advances in Neural Information Processing Systems*, 2017.

M. E. J. Newman and R. M. Ziff. Efficient Monte Carlo algorithm and high-precision results for percolation. *Physical Review Letters*, 85:4104–4107, 2000. doi: 10.1103/PhysRevLett.85.4104.

Nikolas Nüsken and Lorenz Richter. Solving high-dimensional Hamilton–Jacobi–Bellman PDEs using neural networks: perspectives from the theory of controlled diffusions and measures on path space. *Partial Differential Equations and Applications*, 2, 2021. doi: 10.1007/s42985-021-00102-x.

Mathew Penrose. *Random Geometric Graphs*. Oxford University Press, 2003. doi: 10.1093/acprof:oso/9780198506263.001.0001.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *AAAI Conference on Artificial Intelligence*, volume 32, 2018. doi: 10.1609/aaai.v32i1.11671.

Hongbo Qu, X. Wang, Y. Song, W. Ni, G. Jiang, and Q. Sheng. GRLND: A graph reinforcement learning framework for network dismantling. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pp. 2450–2459, 2025. doi: 10.1145/3746252.3761371.

Xiao-Long Ren, Niels Gleinig, Dirk Helbing, and Nino Antulov-Fantulin. Generalized network dismantling. *Proceedings of the National Academy of Sciences*, 116:6554–6559, 2019. doi: 10.1073/pnas.1806108116.

Christian M. Schneider, André A. Moreira, José S. Andrade, Shlomo Havlin, and Hans J. Herrmann. Mitigation of malicious attacks on networks. *Proceedings of the National Academy of Sciences*, 108:3838–3841, 2011. doi: 10.1073/pnas.1009440108.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Dietrich Stauffer and Amnon Aharony. *Introduction to Percolation Theory*. Taylor & Francis, 2nd edition, 1994. doi: 10.1201/9781315274386.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2nd edition, 2018.

Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2): 146–160, 1972. doi: 10.1137/0201010.

Haozhe Tian, Pietro Ferraro, Robert Shorten, Mahdi Jalili, and Homayoun Hamedmoghadam. Learning network dismantling without handcrafted inputs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026. arXiv:2508.00706.

Emanuel Todorov. Efficient computation of optimal actions. *Proceedings of the National Academy of Sciences*, 106(28):11478–11483, 2009. doi: 10.1073/pnas.0710743106.

Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. DiGress: Discrete denoising diffusion for graph generation. In *International Conference on Learning Representations*, 2023.

Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998. doi: 10.1038/30918.

Qitian Wu, Hengrui Zhang, Junchi Yan, and David Wipf. Handling distribution shifts on graphs: An invariance perspective. In *International Conference on Learning Representations*, 2022.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

Dengcheng Yan, Wenxin Xie, Yiwen Zhang, Qiang He, and Yun Yang. Hypernetwork dismantling via deep reinforcement learning. *IEEE Transactions on Network Science and Engineering*, 9(5): 3302–3315, 2022.

Gilad Yehudai, Ethan Fetaya, Eli Meirom, Gal Chechik, and Haggai Maron. From local structures to size generalization in graph neural networks. In *International Conference on Machine Learning*, 2021.

Lenka Zdeborová, Pan Zhang, and Hai-Jun Zhou. Fast and simple decycling and dismantling of networks. *Scientific Reports*, 6:37954, 2016. doi: 10.1038/srep37954.

Bohang Zhang, Shengjie Luo, Liwei Wang, and Di He. Rethinking the expressive power of GNNs via graph biconnectivity. In *International Conference on Learning Representations*, 2023.

Jiazheng Zhang and Bang Wang. Dismantling complex networks by a neural model trained from tiny networks. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management*, pp. 2559–2568, 2022.

Jiazheng Zhang and Bang Wang. Encoding node diffusion competence and role significance for network dismantling. In *Proceedings of the ACM Web Conference*, 2023.

Yu Zheng, Jingtao Ding, Depeng Jin, Jianxi Gao, and Yong Li. Advancing network resilience theories with symbolized reinforcement learning. *arXiv preprint arXiv:2507.08827*, 2025.

A Notation

The accumulated pairwise connectivity of critical node detection (Arulselvan et al., 2009) is

$$\mathcal{P}(\pi; G) = \frac{1}{B+1} \sum_{t=0}^{B} \sum_{C \in \mathcal{C}(S_t)} \binom{|C|}{2} \Big/ \binom{n}{2}, \tag{13}$$

where $\mathcal{C}(S)$ is the set of components of $G[S]$. Table 4 lists the symbols used throughout the paper.

Table 4: Notation.

Symbol	Meaning	Symbol	Meaning
$G = (V, E)$, n, m	input graph; numbers of nodes and edges	r_θ, ℓ_θ, u_θ	learned child log-potential; logit; set log-pot.
S_t, $G[S_t]$	surviving set after t removals; residual graph	q_θ, $\mathbb{P}_\theta$	learned kernel; learned path law
π, B, b_t	trajectory; budget; remaining budget $B - t$	$\xi(v; S)$, τ	articulation sensitivity; structural time
$C_{\max}(S)$, $c(S)$	largest component; $c(S) = \lvert C_{\max}(S)\rvert/n$	$\tilde{\chi}$, Δ^*, ϱ	susceptibility; max. collapse; criticality density
$\mathcal{R}$, B_ρ	AUC objective (Eq. (1)); critical budget	$\gamma(S, b)$	conditioning vector $(\tau, \varrho, \tilde{\chi}, \Delta^*, b/\lvert S\rvert)$
Γ, $\mathcal{P}$	fixed-budget residual; pairwise connectivity	$k(S)$, $k_{\max}$	removals per evaluation; its maximum
$\Phi(\pi)$, $\Phi^\star_B$	path cost $\sum_t c(S_t)$; its minimum	K, T, I, w	samples; temp.; refinement proposals, window
q_0, w, α	reference kernel; weight $(1+\deg)^\alpha$; bias	L, d	message-passing depth; width
$\beta = \bar{\beta} n$, g, ϕ	inverse temp.; running wt. $e^{-\beta c}$; terminal wt.	μ, ϵ_μ	behavior kernel; reference mixing weight
$h_b(S)$, $V_b(S)$	desirability (Eq. (5)); soft cost-to-go	δ_t, δ_t°	TD residual with / without stop-gradient
$q^\star$, $\mathbb{P}^\star$, $\mathbb{Q}_0$	exact kernel (Eq. (6)); path law; reference law	Δ_G, α_{A}	max. degree of G; inverse Ackermann fn.

B Related Work

B.1 Network Dismantling

Network dismantling seeks a small, ordered set of node removals that fragments the giant component. Its performance is commonly summarized by the area under the dismantling curve (Schneider et al., 2011), and Artime et al. (2024) provide a recent review of the field. We group prior work by how much structural knowledge is built into the removal rule.

Centrality-based heuristics remove nodes by static or adaptively recomputed scores such as degree and betweenness (Freeman, 1977; Brandes, 2001) and analyze the resulting percolation transition (Albert et al., 2000; Cohen et al., 2001; Stauffer & Aharony, 1994); related scores such as the k-shell optimize different objectives. Common to all of them is the assumption that a local, recomputed score suffices to rank removals.

Message-passing, spectral, and optimization approaches encode stronger structural priors: Collective Influence approximates optimal percolation through a non-backtracking objective (Morone & Makse, 2015); belief-propagation methods (BPD) reduce dismantling to decycling, tree breaking and reinsertion (Braunstein et al., 2016); CoreHD repeatedly removes the highest-degree node of the 2-core (Zdeborová et al., 2016); generalized network dismantling (GND) uses weighted spectral partitioning (Ren et al., 2019); LGD-NA with its common-neighbor dissimilarity rule (CND) recomputes latent-geometry scores on every test graph (Adler et al., 2026); and reverse dismantling builds the order backwards by reinserting nodes into an empty graph, with a static and a dynamic variant that scale to large graphs (Duan et al., 2026). Reverse dismantling is the training-free method closest to our setting, undirected, unweighted and ordered, and we include its dynamic variant as a baseline (Table 1). Critical node detection is also solved exactly on small instances by integer programming (Arulselvan et al., 2009). These methods transfer by construction, since they are recomputed on every graph, but they inherit the failure modes of their priors, which are typically locally tree-like or sparse-random.

Learning-based dismantling replaces hand-crafted rules with trained models, with the promise of discovering structural roles that no single heuristic captures. FINDER trains a graph neural policy by deep reinforcement learning on small synthetic graphs (Fan et al., 2020), GDM trains a graph network to predict dismantling-relevant node scores (Grassia et al., 2021), MIND learns a dismantling policy without hand-crafted features (Tian et al., 2026), NIRM trains a neural influence-ranking model on tiny synthetic networks and deploys it on large real networks (Zhang & Wang, 2022), NEES learns multiscale node importance with a self-supervised hierarchical encoder (Liu & Wang, 2022), and graph reinforcement learning and unsupervised variants have followed (Qu et al., 2025; Hong et al., 2025). Symbolized reinforcement learning distils the policy of a dismantling agent into interpretable resilience formulas (Zheng et al., 2025); like FINDER and MIND, it trains a policy directly on the dismantling objective, whereas ELDOR computes the immediate structural effect exactly and learns only the budget-dependent future term. Sequential learned dismantling has been extended beyond single-layer graphs: MultiDismantler trains a deep reinforcement learning agent on small synthetic multiplex graphs and transfers it to large real interdependent networks (Gu et al., 2025), and deep reinforcement learning has been applied to hypernetworks with higher-order interactions (Yan et al., 2022). Other work changes the problem itself: heterogeneous removal costs are handled by generalized network dismantling (Ren et al., 2019) and, on multiplex networks, by MultiDismantler (Gu et al., 2025). The unsupervised dismantlers GRLND and GASC (Qu et al., 2025; Hong et al., 2025) also train without labels, but through a surrogate objective or reward, whereas the self-consistency objective of ELDOR enforces a Bellman-type condition on the potential whose zero set identifies the exact Doob kernel (Propositions 9 and 11). NIRM is the closest prior work in the small-to-large transfer regime; it ranks nodes by a learned influence score, whereas ELDOR computes every afterstate cost exactly and learns only a budget-conditioned future term whose exact value defines a known controlled process, and we include NIRM as a baseline (Table 1). NEES likewise produces a node ranking from learned multiscale structure rather than a budget-dependent value. ELDOR assumes an unconstrained, fully observed, uniform-cost single-layer graph. A second line injects richer structure into a learned scorer, for example unsupervised encodings of node diffusion competence and role significance (Zhang & Wang, 2023). Such methods widen what a static score can see; however, the score is still computed once per state, without an exact afterstate cost or a budget-dependent value. Q-learning dismantlers already separate an immediate reward, computed by the environment for the chosen action, from a learned future value. ELDOR differs from them in three respects: the immediate term is an exact afterstate cost computed for *every* candidate in $O(n + m)$ and enters the logits directly; the future term is a soft value bounded by the structure of the problem (Eq. (9)); and, with the exact potential, the resulting kernel samples a known Feynman–Kac path measure whose targets can be checked by dynamic programming on small graphs.

B.2 Neural Combinatorial Optimization and Diffusion

Constructive and heatmap solvers. Autoregressive policies build solutions one element at a time (Kool et al., 2019a); generalization across sizes and distributions remains a known weakness. ELDOR is autoregressive in that it produces a sequence, but its per-step distribution is a Doob-transformed reference kernel rather than an unconstrained policy.

Diffusion-based solvers. Diffusion models (Song et al., 2021) have discrete-state variants, including absorbing processes (Austin et al., 2021) and graph-generative models (Vignac et al., 2023). Diffusion-based solvers for combinatorial problems denoise corrupted indicators of complete solutions and are trained on solver-labeled solutions. ELDOR shares with absorbing diffusion only the forward chain on nested node sets; it has no noise schedule and no learned reverse process. Its time is a structural quantity, its output is an ordered trajectory, and it learns a potential that controls the forward process instead of a denoiser.

B.3 Controlled Processes and Graph Generalization

Doob transforms and controlled path measures. Conditioning a Markov process on a future event through a harmonic function is Doob's h-transform (Doob, 1957); with running costs, the tilted path law is a Feynman–Kac measure (Del Moral, 2004). Linearly solvable MDPs identify the log-desirability with an entropy-regularized value (Todorov, 2009), soft Q-learning learns such values by consistency (Haarnoja et al., 2017), afterstate values exploit transitions that are deterministic given the action (Sutton & Barto, 2018), path-space divergences train controlled processes without target samples (Nüsken & Richter, 2021), and path-consistency learning (Nachum et al., 2017) and GFlowNet objectives (Malkin et al., 2022) enforce related consistency conditions. ELDOR instantiates this machinery on an absorbing, graph-structured state space in which the afterstate cost of every action is exactly computable.

Expressivity and out-of-distribution generalization. Message-passing networks are bounded by the 1-WL test (Xu et al., 2019), higher-order invariant networks are more expressive at a higher cost (Maron et al., 2019), and message passing cannot detect biconnectivity in general (Zhang et al., 2023); depth and width further limit what such networks compute (Loukas, 2020). Size generalization depends on the overlap of local structures between training and test graphs (Yehudai et al., 2021), and invariance-based training addresses distribution shift on graphs more broadly (Wu et al., 2022). ELDOR does not remove these limits. Instead, it supplies exact biconnectivity information outside the message-passing bottleneck and reports where transfer still fails (Appendix M.1).

Summary. To the best of our knowledge, no prior work combines (a) a generative process on irreversible intervention trajectories, (b) a budget-conditioned potential whose exact value yields a known optimal-control path measure, (c) exact non-local afterstate costs outside the message-passing bottleneck, and (d) zero-shot evaluation on 48 real networks from synthetic training alone; Table 5 contrasts these formulations.

Table 5: Conceptual comparison of formulations. "Order": the output is an ordered intervention trajectory by construction.

Formulation	State	Transition	Irreversible	Order	Budget-conditioned
Static / adaptive ranking	node scores	recompute & take top	yes	yes	no
Autoregressive / Q-learning policy	partial solution	learned softmax / argmax	yes	yes	optional
Solution denoising (diffusion)	corrupted solution vector	learned denoiser	no	post hoc	no
ELDOR (ours)	surviving node set S_t	Doob transform of q_0	yes	yes	yes (via h_b)

C Limitations and Scope of the Theory

C.1 Limitations

The immediate term sees what a single removal does. A separator made of several nodes becomes visible to it only after part of the separator has been removed, and the message-passing backbone cannot recover such separators on its own beyond what 1-WL distinguishes. This is the regime in which separator-based heuristics remain ahead, most clearly on the technological networks (Appendix M.1); higher-order message passing, symmetry-breaking inputs or multi-node lookahead could address it, each at an additional cost.

The gains are largest with search. As a single greedy policy, ELDOR is on par with the strongest training-free baselines on average, and restarted LGD-NA closes most of the greedy margin at equal time (Table 16); best-of-16 is significantly better but costs about $30\times$ one heuristic run, and unbatched decoding is quadratic in n for $B = \Theta(n)$ (Table 21). We therefore position ELDOR as a tool for quality-sensitive, offline analysis rather than as a replacement for millisecond heuristics;

distilling best-of-K into the policy, or amortizing the potential across several steps, is the main open problem.

The evaluation covers 48 real networks and one synthetic test protocol, on undirected, unweighted, static, single-layer graphs with uniform removal costs and full observation; it says nothing about network classes these sets do not contain. The released code can regenerate removal sequences for any input network, so withholding precomputed sequences for real infrastructure (see the ethics statement) is a limited safeguard.

C.2 Theoretical Assumptions and Applicability

The theory speaks about three objects that should not be conflated: (A) the exact process driven by $q^\star$, (B) the learned kernel q_θ viewed as a function of its potential, and (C) training and decoding. We state here what is established for each, and under which assumptions.

What is established at each level. *(A) Exact process.* $q^\star$ is a probability kernel with the support of q_0, and the chain it drives has trajectory law $\mathbb{P}^\star \propto \mathbb{Q}_0 e^{-\beta\Phi}\phi$ (Theorem 2); for $\phi \equiv 1$ its soft cost-to-go lies within $\beta^{-1}b(\log n + \alpha\log(1+\Delta_G))$ of the hard one (Proposition 5); and degree rankings are pessimal on the trap family during the first k removals (Proposition 7). *(B) Learned kernel.* For every θ: validity, absorption, budget feasibility and permutation equivariance (Propositions 3 and 4). As a function of the potential: an error of ϵ on all reachable states gives $D_{\mathrm{KL}}(\mathbb{P}_\theta \| \mathbb{P}^\star) \le 2\epsilon B$ (Proposition 8), and a zero consistency residual identifies $q^\star$ (Propositions 9 and 11). *(C) Training and decoding.* Nothing beyond (B) is claimed: neither that the bounded head of Eq. (9) reaches a small residual on large graphs, nor that lagged-target training converges, nor that batched decoding preserves the path law.

Where the premise of the learning guarantee is checked. The premise of Proposition 8 is a uniform bound against $\log h$, which can be evaluated only where $\log h$ is exact. We check it on held-out graphs up to $n = 20$ (Table 17); above that range we report the consistency residual and the agreement with rollout values, which are necessary for the premise but do not imply it. The KL guarantee is therefore not claimed on the test sets.

Terminal weight and head. The path-law statement holds for any $\phi > 0$, whereas the soft-value bounds of Proposition 5 and the bounded head of Eq. (9) use $\phi \equiv 1$, the only parameterization evaluated.

Decoding modes. The path law of Theorem 2 is that of sequential decoding ($k_{\max} = 1$). Frozen-logit batching draws several nodes from one kernel evaluation and is an approximation of that law, not a sample of it; its effect is measured rather than bounded (Table 19). Best-of-K and refinement are search on top of either mode and change the output distribution by design.

Expressivity. The exact inputs supply biconnectivity information that a 1-WL-bounded backbone cannot compute, but they do not make the network universal: nodes that 1-WL cannot distinguish and that have equal exact inputs receive equal logits even when they are not automorphic (Appendix I).

Problem setting. Every exact quantity assumes uniform removal costs on a fully observed single-layer graph. Heterogeneous costs (Ren et al., 2019; Gu et al., 2025) would replace the uniform step of Eq. (1) by a cost-weighted budget and require a cost-aware reference kernel and immediate term; defended nodes and partial observability would make the afterstate cost uncertain, so that the exact term of Eq. (8) could no longer be computed; interdependent layers (Gu et al., 2025) or hyperedges (Yan et al., 2022) would require a different articulation primitive than Lemma 1; and weighted, directed or temporal graphs would change both the reference kernel and the immediate term.

D State Space, Forward Kernel, and Doob-Transform Derivations

Forward kernel. The state at time t is $S_t \in \mathcal{S}_t(G)$. The reference kernel Eq. (3) is a categorical distribution over the $|S_t|$ children. The per-node marginal survival probability under q_0 with $\alpha = 0$ is $\mathbb{Q}_0(v \in S_t) = 1 - t/n$, the linear masking schedule of absorbing discrete diffusion (Austin et al., 2021); for $\alpha > 0$ the marginals are graph-dependent and not available in closed form. The per-node chains are *not* independent: exactly one node is removed per step.

Contrast with solution denoising. A discrete diffusion solver corrupts a solution indicator $x_0 \in \{0,1\}^n$ by independent Bernoulli transitions $q(x_t \mid x_{t-1}) = \prod_i \mathrm{Cat}(x_t^i;\, x_{t-1}^i Q_t)$ and learns $p_\theta(x_{t-1} \mid x_t)$ (Austin et al., 2021). Its intermediate states are arbitrary vectors in $\{0,1\}^n$, reachable from each other in both directions; they do not correspond to intervention histories, the time index is a noise level, and ordering is recovered post hoc. In ELDOR the state space is the Hasse diagram of the subset lattice restricted to downward moves of size one, every state is a partially dismantled graph, and time is tied to structure through τ.

Doob transform with running costs. Let $M(S, S') = q_0(S' \mid S) g(S')$ be the sub-Markov kernel obtained by weighting transitions with $g \le 1$. Equation (5) reads $h_b = M h_{b-1}$, i.e. h_b is space–time harmonic for M: $h_{B-t}(S_t) \prod_{j \le t} g(S_j)$ is a $\mathbb{Q}_0$-martingale in t. The kernel $q^\star(S' \mid S, b) = M(S, S') h_{b-1}(S') / h_b(S)$ is the Doob transform of M by this space–time harmonic function. When $g \equiv 1$ and $\phi = \mathbb{1}_A$, it is the classical conditioning of Doob (1957) in discrete time (Corollary 10). The same construction is known as the optimal twisting function in (controlled) sequential Monte Carlo (Del Moral, 2004) and as the optimal controlled dynamics in linearly solvable MDPs (Todorov, 2009); in reinforcement-learning terms, $\log h_{b-1}(S \setminus \{v\})$ is a soft afterstate value (Sutton & Barto, 2018; Haarnoja et al., 2017).

Why not a time-reversal. Score-based and denoising diffusions learn the time reversal of a noising process. The time reversal of the reference deletion chain is a node-insertion chain from the empty set; learning it would require a data distribution over complete dismantling trajectories, i.e. labels. ELDOR instead learns the *forward* process conditioned on low future cost, which requires only the ability to evaluate c; this is why labels are needed only for the curriculum.

E Additional Formal Statements

The following results are referenced in Section 4; proofs are in Appendix F.

Proposition 3 (Validity, absorption, budget feasibility)**.** *Let q be any kernel satisfying Eq. (2), including q_0, $q^\star$, and q_θ for every θ, and let batched steps draw $k \le b$ distinct nodes of S without replacement. Then every sampled trajectory from (V, B) with $B \le n$ satisfies $V = S_0 \supsetneq S_1 \supsetneq \cdots \supsetneq S_B$, removes exactly B distinct nodes, never reselects a removed node, and stops at $b = 0$ (or at $t^\star \le B$ under threshold stopping).*

Proposition 4 (Permutation equivariance)**.** *Suppose $r_\theta(\sigma v; \sigma S, b, \sigma G) = r_\theta(v; S, b, G)$ for all permutations σ (true for Section 3.2). Then $q_\theta(\sigma v \mid \sigma S, b; \sigma G) = q_\theta(v \mid S, b; G)$, the trajectory law satisfies $\mathbb{P}_\theta(\sigma\pi \mid \sigma G, B) = \mathbb{P}_\theta(\pi \mid G, B)$, also for random-tie greedy, batched Gumbel-top-k, best-of-K, and refinement; h_b is invariant and $q^\star$ equivariant unconditionally.*

Proposition 5 (Soft cost-to-go and temperature bounds)**.** *Let $\phi \equiv 1$, $V_b(S) = -\beta^{-1} \log h_b(S)$, let $V_b^{\mathrm{hard}}(S)$ be the minimum of $\sum_{j=1}^b c(S_{t+j})$ over all b-step continuations from S, Δ_G the maximum degree of G, and $\mathbb{Q}_0^{S,b}$ the reference law of b-step continuations from S. Then:*

(i) $V_b(S) = \min_P \left\{ \mathbb{E}_P[\sum_{j=1}^b c(S_{t+j})] + \beta^{-1} D_{\mathrm{KL}}(P \,\|\, \mathbb{Q}_0^{S,b}) \right\}$ *over distributions P on continuations, with minimizer the restriction of $\mathbb{P}^\star$; equivalently* $V_b(S) = -\beta^{-1} \log \sum_v q_0(v \mid S) e^{-\beta(c(S\setminus\{v\}) + V_{b-1}(S\setminus\{v\}))}$.

(ii) $0 \le V_b^{\mathrm{hard}}(S) \le V_b(S) \le \min\left\{ b\, c(S),\ V_b^{\mathrm{hard}}(S) + \beta^{-1} b \left(\log n + \alpha \log(1 + \Delta_G)\right) \right\}$.

(iii) For $\eta > 0$: $\mathbb{P}^\star\big(\Phi(\pi) \ge \Phi_B^\star + \eta\big) \le e^{-\beta\eta} \left(n(1+\Delta_G)^\alpha\right)^B$.

Lemma 6 (Monotone giant component). *If $S' \subseteq S \subseteq V$ then $|C_{\max}(S')| \le |C_{\max}(S)|$. Hence $c(S_t)$ is nonincreasing and $\tau(S_t)$ nondecreasing along every trajectory.*

Proposition 7 (Degree rankings are pessimal on the trap family during the first k removals). *Let G be generated by Algorithm 2 with k clusters of size s (each connected and d_{in}-regular), one hub per cluster adjacent to h members, and r connectors per edge of a spanning tree over clusters, each adjacent to c members on each side, every member adjacent to at most one connector, and $\max\{d_{\text{in}}+2,\, 2c\} < h$. Then: (a) hubs have degree h, members degree in $\{d_{\text{in}}, d_{\text{in}}+1, d_{\text{in}}+2\}$, connectors degree $2c$; (b) for any set H' of hubs, $G - H'$ is connected; (c) degree and adaptive-degree removal (any tie-breaking) remove the k hubs first, and satisfy $|C_{\max}(S_t)| = n - t$ for $t \le k$, the largest value attainable by any trajectory; (d) removing all $r(k-1)$ connectors leaves components of size $s+1$.*

Proposition 8 (Potential error to path error). *Assume $|r_\theta(v; S, b) - \log h_{b-1}(S \setminus \{v\})| \le \epsilon$ for all reachable (S, v, b), with $\phi \equiv 1$. Then $|\log q_\theta(v \mid S, b) - \log q^\star(v \mid S, b)| \le 2\epsilon$, $D_{\mathrm{KL}}(\mathbb{P}_\theta \| \mathbb{P}^\star) \le 2\epsilon B$, $\mathrm{TV}(\mathbb{P}_\theta, \mathbb{P}^\star) \le \sqrt{\epsilon B}$, and $|\mathbb{E}_{\mathbb{P}_\theta}\Phi - \mathbb{E}_{\mathbb{P}^\star}\Phi| \le B\sqrt{\epsilon B}$.*

Proposition 9 (Consistency characterizes exactness). *Let $u_\theta(S, b) = \mathrm{LSE}_v\, \ell_\theta(v; S, b)$ with $u_\theta(\cdot, 0) = \log \phi$. (i) $q_\theta(\cdot \mid S, b) = q^\star(\cdot \mid S, b)$ iff $r_\theta(v; S, b) - \log h_{b-1}(S \setminus \{v\})$ does not depend on $v \in S$. (ii) If the residual $r_\theta(v; S, b) - u_\theta(S \setminus \{v\}, b-1)$ vanishes for all states reachable from (V, B), then $u_\theta = \log h$ and $q_\theta = q^\star$ on those states. (iii) For any strictly positive state weighting, the set of parameters attaining zero $\mathcal{L}_{\mathrm{TD}}$ under a full-support behavior kernel is the same.*

Corollary 10 (Classical Doob conditioning). *If $g \equiv 1$ and $\phi = \mathbb{1}_A$ for a collapse set $A = \{S : |C_{\max}(S)| \le \rho n\}$ with $\mathbb{Q}_0(S_B \in A) > 0$, then $h_b(S) = \mathbb{Q}_0(S_{t+b} \in A \mid S_t = S)$, and on states with $h_b(S) > 0$ the kernel Eq. (6) is the law of the reference chain conditioned on $\{S_B \in A\}$. Transitions into states with $h_{b-1} = 0$ receive probability 0, so here the support of $q^\star$ is a subset of that of q_0.*

Proposition 11 (Trajectory residual). *For every trajectory π from (V, B), $\sum_{t=0}^{B-1} \delta_t^\circ = \log \mathbb{P}_\theta(\pi) - \log \mathbb{P}^\star(\pi) + u_\theta(V, B) - \log h_B(V)$. Consequently, if μ has full support and $\mathcal{L}_{\mathrm{TB}} = 0$, then $\mathbb{P}_\theta = \mathbb{P}^\star$ and $u_\theta(V, B) = \log h_B(V)$; conversely, both conditions together give $\mathcal{L}_{\mathrm{TB}} = 0$.*

Proposition 12 (Complexity). *Algorithm 4 runs in $O\big(KT_{\mathrm{ev}}(L(n+m)d + Lnd^2) + (K+I)(n+m)\alpha_{\mathrm{A}}(n)\big)$ time with $\lceil B/k_{\max} \rceil \le T_{\mathrm{ev}} \le B$, and $O(n+m+nd)$ memory per concurrently evaluated sample. The exact DP of Appendix G runs in $O(2^n(n+m))$ time and $O(2^n)$ memory per graph and budget.*

F Full Proofs

Throughout, G is finite, simple, undirected; $B \le n$; $\beta > 0$; $g = e^{-\beta c}$; $\phi > 0$ unless stated. "Reachable" means reachable from (V, B) with positive $\mathbb{Q}_0$-probability; by full support of q_0 this is every (S, b) with $|S| = n - (B - b)$.

Proof of Lemma 6. Every connected component C' of $G[S']$ is connected in $G[S] \supseteq G[S']$ and so lies inside one component C of $G[S]$; hence $|C'| \le |C| \le |C_{\max}(S)|$. Along a trajectory $S_{t+1} \subset S_t$, so $c(S_{t+1}) \le c(S_t)$ and $\tau(S_{t+1}) \ge \tau(S_t)$. □

Proof of Lemma 1. Compute the components of $G[S]$ and their sizes by BFS in $O(|S| + |E_S|)$ and record the largest and second-largest sizes $M_1 \ge M_2$ (with $M_2 = 0$ if there is one component). Run one iterative DFS per component K with discovery times disc, low-points $\mathrm{low}(w) = \min\{\mathrm{disc}(w),\ \mathrm{disc}(x) : x \text{ reachable from } T_w \text{ by one back edge}\}$, and subtree sizes $|T_w|$ (Tarjan, 1972). Fix $v \in K$. *Case 1: v is not the DFS root.* Let W_v be the set of DFS children w of v with $\mathrm{low}(w) \ge \mathrm{disc}(v)$. For $w \in W_v$, every edge leaving T_w other than the tree edge $\{v, w\}$ ends in T_w or at v (a back edge from T_w to a proper ancestor of v would give $\mathrm{low}(w) < \mathrm{disc}(v)$; undirected DFS has no cross edges), so T_w is a component of $G[S \setminus \{v\}]$. The remaining nodes $R = K \setminus (\{v\} \cup \bigcup_{w \in W_v} T_w)$ contain the parent side $K \setminus T_v$, which is nonempty and connected (it contains the root and is closed under taking DFS parents), and every $T_{w'}$ with $w' \notin W_v$, each connected by a back edge to a proper ancestor of v, i.e. to $K \setminus T_v$. Hence R is one component, of size $|K| - 1 - \sum_{w \in W_v} |T_w|$. *Case 2: v is the root.* There are no back edges between distinct child subtrees, so the pieces are exactly $\{T_w\}$ over the children w. In both cases the largest piece $p(v)$

and the number of pieces follow in time proportional to the number of DFS children of v. Finally, $|C_{\max}(S \setminus \{v\})| = \max\{p(v), M_{\neq K}\}$, where $M_{\neq K} = M_2$ if $|K| = M_1$ and K is the (chosen) largest component, and M_1 otherwise; if $K = \{v\}$ then $p(v) = 0$. Summing over v, the total work is $O(|S| + |E_S|)$. The same pieces give $\mathcal{P}$ for every child in the same time. □

Proof of Proposition 3. By induction on t: $S_0 = V$. If $S_t \subseteq V$ with $|S_t| = n - t$ and $t < B$, a single step draws v with $q(v \mid S_t, b_t) > 0$, which by Eq. (2) implies $v \in S_t$; then $S_{t+1} = S_t \setminus \{v\}$ has $|S_{t+1}| = n - t - 1$. A batch step of size $k \le b_t \le |S_t|$ draws k distinct elements of S_t without replacement and removes them in order, which is k single steps. A node removed at step t is not in $S_{t'}$ for $t' \ge t$ and hence has probability 0 thereafter. The process performs single-node removals until $t = B$ (or earlier stopping), so exactly B (respectively $t^\star$) distinct nodes are removed. □

Proof of Proposition 4. Degree, component membership and sizes, low-points-derived piece sizes, and ξ are functions of the isomorphism class of $(G[S], v)$, so the node inputs satisfy $x_{\sigma v}(\sigma S; \sigma G) = x_v(S; G)$ and the global quantities γ, c, ϱ are invariant. Message passing with symmetric aggregation over neighbors and symmetric pooling over S is equivariant (Maron et al., 2019), so z_θ and hence r_θ are equivariant, and $\ell_\theta(\sigma v; \sigma S, b; \sigma G) = \ell_\theta(v; S, b; G)$ (note $\log q_0$ and $c(S \setminus \{v\})$ are equivariant too). The softmax over S is equivariant, giving the kernel statement. For trajectories, $\mathbb{P}_\theta(\pi \mid G, B) = \prod_t q_\theta(v_t \mid S_{t-1}, b_{t-1}; G)$, and each factor is unchanged under $(\pi, G) \mapsto (\sigma\pi, \sigma G)$ because σS_{t-1} is the surviving set of $\sigma\pi$. Greedy decoding with uniformly random tie-breaking selects uniformly among $\arg\max \ell_\theta$, an equivariant set; Gumbel-top-k applies i.i.d. noise to equivariant logits, and $k(S)$ is invariant; best-of-K selects by the invariant $\mathcal{R}$ with uniform tie-breaking; window refinement proposes moves by position, and positions are preserved by relabeling. h_b is invariant by induction on Eq. (5) (all terms are), hence $q^\star$ is equivariant. □

Proof of Theorem 2. (i) $\sum_{v \in S} q^\star(v \mid S, b) = h_b(S)^{-1} \sum_v q_0(v \mid S) g(S \setminus \{v\}) h_{b-1}(S \setminus \{v\}) = 1$ by Eq. (5). Positivity of h follows by induction from $\phi > 0$, $g > 0$, and full support of q_0 on nonempty S (note $b \le |S|$ ensures $S \neq \emptyset$ whenever $b \ge 1$), so the ratio is well defined. (ii) For $v \in S$ all factors are strictly positive; for $v \notin S$, $q_0(v \mid S) = 0$. (iii) For $\pi = (v_1, \dots, v_B)$,

$$\prod_{t=1}^{B} q^\star(v_t \mid S_{t-1}, B{-}t{+}1) = \prod_{t=1}^{B} q_0(v_t \mid S_{t-1}) g(S_t) \cdot \prod_{t=1}^{B} \frac{h_{B-t}(S_t)}{h_{B-t+1}(S_{t-1})} = \mathbb{Q}_0(\pi) e^{-\beta\Phi(\pi)} \frac{h_0(S_B)}{h_B(S_0)},$$

which is Eq. (7) since $h_0 = \phi$ and $S_0 = V$. □

Proof of Corollary 10. With $g \equiv 1$, Eq. (5) is the backward equation for $h_b(S) = \mathbb{Q}_0(S_{t+b} \in A \mid S_t = S)$ (time-homogeneous kernel). For $h_b(S) > 0$, Bayes' rule and the Markov property give $\mathbb{Q}_0(S_{t+1} = S \setminus \{v\} \mid S_t = S, S_B \in A) = q_0(v \mid S) h_{b-1}(S \setminus \{v\}) / h_b(S)$, which is Eq. (6). The last claim is immediate. □

Lemma 13 (Unrolled desirability). $h_b(S) = \mathbb{E}_{\mathbb{Q}_0}\big[\prod_{j=1}^{b} g(S_{t+j})\, \phi(S_{t+b}) \,\big|\, S_t = S\big]$.

Proof. Induction on b: true for $b = 0$; conditioning on the first step and using the Markov property gives Eq. (5). □

Proof of Proposition 5. Let Ω be the finite set of b-step continuations from S, $Q = \mathbb{Q}_0^{S,b}$ (full support on Ω), and $f(\omega) = \beta \sum_{j=1}^{b} c(S_{t+j})$. For any distribution P on Ω define $P_f = Q e^{-f}/Z$ with $Z = \sum_\omega Q(\omega) e^{-f(\omega)} = h_b(S)$ (Lemma 13 with $\phi \equiv 1$). Then $\mathbb{E}_P f + D_{\mathrm{KL}}(P \| Q) = D_{\mathrm{KL}}(P \| P_f) - \log Z \ge -\log Z$ with equality iff $P = P_f$. Dividing by β gives (i); P_f is the restriction of $\mathbb{P}^\star$ by Theorem 2. The recursive form follows from Eq. (5) after taking $-\beta^{-1} \log$. (ii) $V_b \ge 0$ because $f \ge 0$ and $D_{\mathrm{KL}} \ge 0$; $V_b \ge V_b^{\mathrm{hard}}$ because $\mathbb{E}_P f/\beta \ge V_b^{\mathrm{hard}}$ for every P. Choosing $P = Q$ gives $V_b \le \mathbb{E}_Q[\sum_j c(S_{t+j})] \le b\, c(S)$ by Lemma 6. Choosing $P = \delta_{\omega^\star}$ for a minimizing continuation gives $V_b \le V_b^{\mathrm{hard}} + \beta^{-1} \log(1/Q(\omega^\star))$, and $q_0(v \mid S') \ge 1/(|S'|(1 + \Delta_G)^\alpha)$ because $w \ge 1$ and $\sum_u w(u; S') \le |S'|(1 + \Delta_G)^\alpha$, so $\log(1/Q(\omega^\star)) \le b(\log n + \alpha \log(1 + \Delta_G))$. (iii) With $\phi \equiv 1$ and $\pi^\star$ optimal, $\mathbb{P}^\star(\Phi \ge \Phi_B^\star + \eta) = \sum_{\Phi(\pi) \ge \Phi_B^\star + \eta} \mathbb{Q}_0(\pi) e^{-\beta\Phi(\pi)} / h_B(V) \le e^{-\beta(\Phi_B^\star + \eta)} / (\mathbb{Q}_0(\pi^\star) e^{-\beta\Phi_B^\star})$, using $\sum \mathbb{Q}_0 \le 1$ and $h_B(V) \ge \mathbb{Q}_0(\pi^\star) e^{-\beta\Phi_B^\star}$; the bound on $1/\mathbb{Q}_0(\pi^\star)$ from (ii) completes the proof. □

Proof of Proposition 9. (i) $q_\theta(v) \propto a_v e^{r_\theta(v)}$ and $q^\star(v) \propto a_v h_{b-1}(S \setminus \{v\})$ with $a_v = q_0(v \mid S)g(S \setminus \{v\}) > 0$ on S. Two such distributions coincide iff $e^{r_\theta(v)}/h_{b-1}(S \setminus \{v\})$ is constant in v. (ii) Induction on b. For $b = 0$, $u_\theta(\cdot, 0) = \log \phi = \log h_0$. If $u_\theta(\cdot, b-1) = \log h_{b-1}$ on reachable states, the vanishing residual gives $r_\theta(v; S, b) = \log h_{b-1}(S \setminus \{v\})$, and then $u_\theta(S, b) = \mathrm{LSE}_v[\log q_0(v \mid S) - \beta c(S \setminus \{v\}) + \log h_{b-1}(S \setminus \{v\})] = \log h_b(S)$ by Eq. (5); $q_\theta = q^\star$ follows from (i). (iii) At a fixed point $\bar\theta = \theta$, $\mathcal{L}_{\mathrm{TD}} = 0$ iff the residual vanishes μ-almost surely; with a full-support μ this is every reachable transition, a condition that does not involve the (strictly positive) weights. □

Proof of Proposition 11. From Eq. (8), $\log q_\theta(v \mid S, b) = \log q_0(v \mid S) - \beta c(S \setminus \{v\}) + r_\theta(v; S, b) - u_\theta(S, b)$. Summing along π and subtracting $\log \mathbb{P}^\star(\pi) = \sum_t [\log q_0(v_{t+1} \mid S_t) - \beta c(S_{t+1})] + \log \phi(S_B) - \log h_B(V)$ yields $\log \mathbb{P}_\theta - \log \mathbb{P}^\star = \sum_{t=0}^{B-1} [r_\theta(v_{t+1}; S_t, b_t) - u_\theta(S_t, b_t)] - \log \phi(S_B) + \log h_B(V)$. On the other hand $\sum_t \delta_t^\circ = \sum_t r_\theta(v_{t+1}; S_t, b_t) - \sum_{t=1}^{B} u_\theta(S_t, b_t)$, and $\sum_{t=1}^{B} u_\theta(S_t, b_t) = \sum_{t=0}^{B-1} u_\theta(S_t, b_t) - u_\theta(V, B) + \log \phi(S_B)$. Combining gives the identity. If $\mathcal{L}_{\mathrm{TB}} = 0$ under full-support μ, then $\log \mathbb{P}_\theta(\pi) - \log \mathbb{P}^\star(\pi) = \log h_B(V) - u_\theta(V, B)$ is a constant κ for every π; summing $\mathbb{P}_\theta = e^\kappa \mathbb{P}^\star$ over π gives $\kappa = 0$. The converse follows from the identity. □

Proof of Proposition 8. Write $r^\star(v) = \log h_{b-1}(S \setminus \{v\})$ and a_v as above. Then $\log q_\theta(v) - \log q^\star(v) = (r_\theta(v) - r^\star(v)) - \log \frac{\sum_u a_u e^{r_\theta(u)}}{\sum_u a_u e^{r^\star(u)}}$. The first term lies in $[-\epsilon, \epsilon]$; since $e^{-\epsilon} e^{r^\star(u)} \le e^{r_\theta(u)} \le e^\epsilon e^{r^\star(u)}$, so does the log-ratio of normalizers; hence the difference is in $[-2\epsilon, 2\epsilon]$. The chain rule gives $D_{\mathrm{KL}}(\mathbb{P}_\theta \| \mathbb{P}^\star) = \sum_t \mathbb{E}_{\mathbb{P}_\theta}[D_{\mathrm{KL}}(q_\theta(\cdot \mid S_t, b_t) \| q^\star(\cdot \mid S_t, b_t))]$, and each term is at most $\sup |\log q_\theta - \log q^\star| \le 2\epsilon$. Pinsker's inequality gives $\mathrm{TV} \le \sqrt{D_{\mathrm{KL}}/2} \le \sqrt{\epsilon B}$, and since $\Phi \in [0, B]$, $|\mathbb{E}_{\mathbb{P}_\theta} \Phi - \mathbb{E}_{\mathbb{P}^\star} \Phi| \le B\,\mathrm{TV}$. □

Proof of Proposition 7. (a) A hub is adjacent to exactly h members and nothing else. A member has d_{in} cluster neighbors, at most one hub, and at most one connector. A connector has c members on each side. (b) Each cluster is connected; each hub attaches to its cluster; each connector attaches to two clusters adjacent in a spanning tree; hence G is connected, and hubs are not needed for any of these connections, so $G - H'$ remains connected. (c) Hubs are pairwise non-adjacent and non-adjacent to connectors, so a hub's degree stays h until it is removed, whereas every other degree is initially at most $\max\{d_{\mathrm{in}} + 2, 2c\} < h$ and can only decrease under deletions. Static and adaptive degree therefore remove the k hubs first, under any tie-breaking among hubs. By (b), $G[S_t]$ is connected for $t \le k$, so $|C_{\max}(S_t)| = |S_t| = n - t$, and $|C_{\max}(S_t)| \le |S_t| = n - t$ for any trajectory. (d) Without connectors, the components are exactly cluster ∪ hub, of size $s + 1$. □

Proof of Proposition 12. Per network evaluation: the articulation pass is $O(n+m)$ (Lemma 1); each of L layers aggregates over $O(m)$ edges with width d and applies node MLPs in $O(nd^2)$; the head and softmax are $O(nd)$ and $O(n)$; Gumbel-top-k is $O(n)$ with selection. There are T_{ev} evaluations per sample and K samples. Reverse union–find evaluation of one trajectory adds nodes and their edges once each, $O((n + m)\alpha_{\mathrm{A}}(n))$; each refinement proposal re-evaluates once. Memory holds the graph, the mask, and $O(nd)$ activations per layer (no activations are retained at inference). The DP visits each subset once and performs one articulation pass per subset. □

G Exact Dynamic Programming for Small Graphs

Cost: $\sum_{j=n-B}^{n} \binom{n}{j} O(n + m) \le O(2^n(n + m))$ time and $O(2^n)$ memory in double precision. For $n = 20$ this is about 10^6 subsets per (G, B).

H Degree-Trap Generator

Default parameters (protocol): $(k, s) \in \{(8, 50), (16, 100)\}$, $d_{\mathrm{in}} = 4$, $h = 30$, $c = 2$ (connector degree 4, inside the member degree range $\{4, 5, 6\}$), $r \in \{1, 2, 3\}$; both pairs satisfy $s \ge c\,r\,(k-1)$ for $r \le 3$ ($42 \le 50$ and $90 \le 100$). The node count is $n = k(s + 1) + r(k - 1) \in [415, 1{,}661]$. Roles are never given to any method.

Algorithm 1 Exact soft and hard cost-to-go on a small graph

Require: G with $n \le n_{\mathrm{ex}}$, budget B, β, α
represent $S \subseteq V$ as a bitmask; $b(S) = B - (n - |S|)$
for S with $|S| = n - B$ **do**
  $\log h(S) \leftarrow 0$; $V^{\mathrm{hard}}(S) \leftarrow 0$
for $j = n - B + 1, \dots, n$ **do** ▷ increasing $|S|$
  for S with $|S| = j$ **do**
    compute $c(S \setminus \{v\})$ for all $v \in S$ (Lemma 1); $\log q_0(v \mid S)$ from Eq. (3)
    $\log h(S) \leftarrow \mathrm{LSE}_{v \in S}[\log q_0(v \mid S) - \beta c(S \setminus \{v\}) + \log h(S \setminus \{v\})]$
    $V^{\mathrm{hard}}(S) \leftarrow \min_{v \in S}[c(S \setminus \{v\}) + V^{\mathrm{hard}}(S \setminus \{v\})]$
return tables $\log h$, V^{hard}; $\Phi_B^\star = V^{\mathrm{hard}}(V)$; $q^\star$ via Eq. (6)

Algorithm 2 Degree-trap generator $\textsc{Trap}(k, s, d_{\mathrm{in}}, h, r, c)$

Require: $k \ge 2$, $s \ge \max\{h,\, c\, r\,(k-1)\}$, $d_{\mathrm{in}} \ge 3$ with $s\, d_{\mathrm{in}}$ even, $\max\{d_{\mathrm{in}} + 2, 2c\} < h \le s$, $r \ge 1$, $c \ge 1$
repeat
  for $i = 1, \dots, k$ **do**
    $K_i \leftarrow$ random d_{in}-regular graph on s new nodes, resampled until connected
  for $i = 1, \dots, k$ **do**
    add hub η_i adjacent to h uniformly random members of K_i
  $T \leftarrow$ uniform random labeled tree on $\{1, \dots, k\}$ (Prüfer sequence)
  for each edge $\{i, j\} \in T$ and $\ell = 1, \dots, r$ **do**
    add connector $x_{ij\ell}$ adjacent to c members of K_i and c members of K_j, sampled uniformly among members with no connector neighbor yet
until $r = 1$ **or** G has no articulation point ▷ checked in $O(n + m)$
randomly permute node labels; **return** G and (for evaluation only) the role of each node

I Architecture Details

Inputs. At each state the network receives only $G[S]$, b, and $\gamma(S, b)$. Node inputs are recomputed from $G[S]$: $x_v = \big(\log(1 + \deg_S v),\ \log(1 + \deg_S v) - \overline{\log(1 + \deg_S)}_{N_S(v)},\ \xi(v; S),\ \log(1 + \#\text{pieces}(v; S)),\ \mathbb{1}[v \in C_{\max}(S)],\ |C_S(v)|/|C_{\max}(S)|\big)$, where $C_S(v)$ is v's component. All are ratios or logarithms, and none depends on node identity or on n directly.

Backbone. L residual message-passing layers on $G[S]$ (removed nodes are absent, not masked features): $\mathbf{e}_v^{(\ell+1)} = \mathbf{e}_v^{(\ell)} + \big(1 + a_\ell(\gamma)\big) \odot \mathrm{MLP}_\ell\big(\mathrm{LN}[\mathbf{e}_v^{(\ell)}, \mathrm{mean}_{u \in N_S(v)} \mathbf{e}_u^{(\ell)}, \max_{u \in N_S(v)} \mathbf{e}_u^{(\ell)}]\big) + b_\ell(\gamma)$, with FiLM conditioning (a_ℓ, b_ℓ) (Perez et al., 2018), and a global readout $\bar{\mathbf{e}} = [\mathrm{mean}_{v \in S} \mathbf{e}_v^{(L)}, \max_{v \in S} \mathbf{e}_v^{(L)}]$. Mean/max aggregation avoids sum-aggregation scale drift across sizes (Yehudai et al., 2021).

Expressivity (full discussion). The backbone is at most as discriminative as 1-WL (Xu et al., 2019), and 1-WL-bounded networks cannot in general detect cut vertices (Zhang et al., 2023). The exact inputs ξ and piece counts supply this biconnectivity information from outside the message-passing bottleneck, but they do not make the network universal: nodes that 1-WL cannot distinguish and that have equal exact inputs receive equal logits, even when they are not automorphic. We do not add random node features by default, because they would make the kernel equivariant only in distribution; this is an ablation.

Numerics. Logits ℓ_θ scale like $\beta(b-1)c = \bar{\beta}(b-1)|C_{\max}|$, which can reach 10^8 or more for large graphs. We (i) evaluate Eq. (9) and the softmax in float64 with max-subtraction; (ii) divide logits by $\bar{\beta}T$ before sampling (Section 3.3), so that sampling diversity does not vanish as n grows; (iii) normalize TD residuals by s_t (Eq. (12)) so that the loss is scale-free; (iv) anneal $\bar{\beta}$ upward during training; (v) monitor saturation of $\sigma(z_\theta)$. If saturation impedes learning, an alternative head predicts a state baseline and a bounded per-node advantage, $r_\theta = -\beta[(b-1)c(S)\sigma(\bar{z}_\theta(S)) + (b-1)c(S)\tanh(a_\theta(v))/2]$, whose softmax is invariant to the baseline; this variant is included in Table 9 as a search option.

J Training Objectives Considered, Training and Inference Algorithms

(a) Exact DP supervision is available only for $n \le n_{\text{ex}}$. (b) Imitation of heuristic solvers imports their biases, which is precisely what we want to avoid under shift. (c) Policy gradient on $\mathcal{R}$ is unbiased but high-variance for long trajectories. (d) Path-space divergences and (e) temporal-difference consistency train the potential without labels. We choose a single primary objective—*soft-Bellman consistency* of the potential, i.e. (d)+(e)—and use (a) as a curriculum on small graphs; (c) is an optional fine-tuning stage that we report separately.

Algorithm 3 ELDOR training (exact curriculum + unlabeled consistency)

Require: generator pool $\mathfrak{F}_{\text{tr}}$, sizes $\mathcal{N}_{\text{tr}}$, exact pool $\mathcal{G}_{\text{ex}}$ ($n \le n_{\text{ex}}$) with tables $\log h$, schedules $\lambda_{\text{ex}}(i), \bar{\beta}(i), n_{\max}(i)$

for $i = 1, \dots, N_{\text{iter}}$ **do**
- draw $G \in \mathcal{G}_{\text{ex}}$, budget B; roll out μ from V until B or collapse; compute $\mathcal{L}_{\text{ex}}$ on visited non-collapsed (S, b) ▷ Eq. (10)
- draw $G \sim \mathfrak{F}_{\text{tr}}$ with $n \le n_{\max}(i)$, $B \sim \text{Unif}$; roll out μ with $k(S) \equiv 1$ until B or collapse
- set $u(S, b) \leftarrow 0$ for collapsed S and $u(\cdot, 0) \leftarrow 0$ ▷ terminal convention, Appendix J.1
- push $(S_t, b_t, v_{t+1}, S_{t+1}, \varrho(S_t))$ to replay buffer $\mathcal{B}$
- sample minibatch from $\mathcal{B}$; compute $\mathcal{L}_{\text{TD}}$ with target net $\bar{\theta}$ ▷ Eq. (12)
- compute $\mathcal{L}_{\text{TB}}$ on the fresh trajectory
- $\theta \leftarrow \theta - \eta \nabla_\theta \big(\lambda_{\text{ex}}(i)\mathcal{L}_{\text{ex}} + \mathcal{L}_{\text{TD}} + \lambda_{\text{TB}}\mathcal{L}_{\text{TB}}\big)$; $\bar{\theta} \leftarrow \kappa\bar{\theta} + (1-\kappa)\theta$

select the checkpoint by mean $\mathcal{R}$ on the synthetic validation cells only

Algorithm 4 ELDOR inference

Require: G, budget B, optional threshold ρ, samples K, temperature T, $k_{\max}$, refinement budget I

for $j = 1, \dots, K$ **do** ▷ $j = 1$ greedy; $j > 1$ stochastic
- $S \leftarrow V$, $\pi_j \leftarrow (\,)$, $t \leftarrow 0$
- **while** $t < B$ **and not** (ρ given **and** $|C_{\max}(S)| \le \rho n$) **do**
  - articulation pass (Lemma 1): $c(S \setminus \{v\})\ \forall v$, γ, ϱ; batch size $k \leftarrow k(S)$ (Section 3.1)
  - one network pass: $\ell_v \leftarrow \log q_0(v \mid S) - \beta c(S \setminus \{v\}) + r_\theta(v; S, B-t)$, $v \in S$
  - $u_{1:k} \leftarrow$ top-k of ℓ ($j = 1$) or Gumbel-top-k of $\ell/(\bar{\beta}T)$; append to π_j; remove from S; $t \leftarrow t + k$

$\pi^\star \leftarrow \arg\min_j \mathcal{R}(\pi_j)$ (reverse union–find)

optionally apply I window moves to $\pi^\star$, accepting a move only if it lowers $\mathcal{R}$; **return** $\pi^\star$

Algorithm 3 combines (a) with (d)+(e). A policy-gradient stage, if used, optimizes $\mathbb{E}_{\mathbb{P}_\theta}[\mathcal{R}]$ with a greedy-rollout baseline (Kool et al., 2019a) and is reported as a separate row, never merged into the primary model.

J.1 Zero-Residual Condition and Post-Collapse States

From the recursion to the losses. Taking logarithms in Eq. (5) and using the logits of Eq. (8),

$$\log h_b(S) = \text{LSE}_{v \in S}\big[\log q_0(v \mid S) - \beta c(S \setminus \{v\}) + \log h_{b-1}(S \setminus \{v\})\big],$$
$$u_\theta(S, b) = \text{LSE}_{v \in S}\big[\log q_0(v \mid S) - \beta c(S \setminus \{v\}) + r_\theta(v; S, b)\big].$$

Hence, if $r_\theta(v; S, b) = u_\theta(S \setminus \{v\}, b-1)$ for every v and b (the *zero-residual condition*), then by induction on b from $u_\theta(\cdot, 0) = \log\phi = \log h_0$ we get $u_\theta = \log h$ and $r_\theta = \log h_{b-1}$, so $q_\theta = q^\star$ (Proposition 9). The TD residual of Eq. (11) is this condition evaluated at the sampled removal, $\delta_t = r_\theta(v_{t+1}; S_t, b_t) - u_{\bar{\theta}}(S_{t+1}, b_t - 1)$. Summing the undelayed residuals δ_t° along a trajectory and using $\log q_\theta(v_{t+1} \mid S_t, b_t) = \ell_\theta(v_{t+1}; S_t, b_t) - u_\theta(S_t, b_t)$ gives the trajectory identity

$$\textstyle\sum_{t=0}^{B-1} \delta_t^\circ = \log \mathbb{P}_\theta(\pi) - \log \mathbb{Q}_0(\pi) + \beta\Phi(\pi) + u_\theta(V, B) - \log\phi(S_B),$$

which vanishes for every π exactly when $\mathbb{P}_\theta = \mathbb{P}^\star$ and $u_\theta(V, B) = \log h_B(V)$ (Proposition 11).

Post-collapse states. If $c(S) = 0$, i.e. $|C_{\max}(S)| \le \rho n$, then $c(S') = 0$ for every $S' \subseteq S$ (Lemma 6), so $g \equiv 1$ below S and, with $\phi \equiv 1$, $h_b(S) = 1$ and $q^\star(\cdot \mid S, b) = q_0(\cdot \mid S)$ for every b. Such states carry no learning signal: they are excluded from $\mathcal{D}_{\text{ex}}$, rollouts stop at the first collapsed state, TD targets use $u(S, b) = 0$ for collapsed S, and every normalizer is floored at βn^{-1}, as in s_{ex} and s_t.

K Full Experimental Protocol

This section describes the data, splits, baselines and statistics; Table 6 summarizes the splits and Table 7 lists the parameter ranges.

Table 6: Dataset and split protocol. Training and validation cells are disjoint, and both test sets are those of the real networks, used for zero-shot testing only; parameter ranges are listed in Table 7.

Split	Families	n	Parameters	Graphs
Train (unlabeled)	ER, BA, WS, SBM, RGG, PPM, frag.	200–1,000	training cells	9,600
Train (exact labels)	same seven families	8–20	training cells	2,100
Validation	training families	1,000–1,500	held-out cells	1,200
Test (Synthetic protocol)	ER, CM, SBM	10^3–10^5	CM unseen; $n > 10^3$	90
Test (mechanism)	fragmentation, held-out	$\approx 10^3$	$w \geq 3$, symm., cuts	240
Test (real)	real networks, four domains	128–1,402,673	—	48

Training data. ELDOR and Structural-AR are trained on the same pool of seven generator families: Erdős–Rényi (Erdős & Rényi, 1959), Barabási–Albert (Barabási & Albert, 1999), Watts–Strogatz (Watts & Strogatz, 1998), stochastic block models (Holland et al., 1983), random geometric graphs (Penrose, 2003), planted-partition graphs (the equal-block special case of the SBM, kept as a separate family) and fragmentation graphs (Appendix H): 9,600 unlabeled graphs with $n \in [200, 1{,}000]$ (1,400 per family and 1,200 fragmentation graphs) and 2,100 exact-labeled graphs with $n \in [8, 20]$ (300 per family). FINDER, GDM and MIND are not retrained: their published results are quoted, and for runtime and equal-time comparisons MIND is run from its released checkpoint.

Table 7: Parameter ranges of the generator suite. Validation cells are disjoint from the training cells.

Parameter	Training cells	Validation cells
Node count n	200–1,000	1,000–1,500
Mean degree $\langle k \rangle$	$[2.5, 8]$	$(8, 11]$
Global clustering C	$[0.01, 0.45]$	$(0.45, 0.6]$
Communities q	$[2, 8]$	$(8, 12]$
Community imbalance ι	$[0, 0.6]$	$(0.6, 0.8]$
Intra/inter density $p_{\text{in}}/p_{\text{out}}$	$[5, 40]$	$(40, 80]$
Separator width w	$\{1, 2\}$	$\{3\}$
Connector degree κ	$[2, 6]$	$(6, 9]$
Decoy-hub count	$\{4, 8\}$	$\{12\}$
Noise-edge fraction η	$[0, 0.02]$	$(0.02, 0.05]$
Budget B/n	$[0.1, 0.5]$	$(0.5, 0.6]$
Collapse threshold ρ	$\{0.05, 0.1\}$	$\{0.02\}$

Validation data. Validation graphs come from the training families but from held-out parameter cells (Table 7); they are the only data used for checkpoint selection, early stopping and hyperparameter search. No test network is used for any decision.

Test data. Both test sets are used for zero-shot testing only: the 48 real networks with 128 to 1,402,673 nodes, and the synthetic protocol of Erdős–Rényi, configuration-model and SBM graphs with 10^3 to 10^5 nodes (Appendix L). For ELDOR, the configuration-model family is unseen and every size above 10^3 exceeds the largest training graph; the ER and SBM cells at 10^3 share family and size range with training. All learned methods use the decoder configuration $k_{\max} = 16$ (Table 9). Access conditions and licenses are those of the originating repositories.

Baselines and tuning. The nine published methods (adaptive degree, betweenness, PageRank, min-sum, explosive immunization, GND, FINDER, GDM and MIND) are quoted from their published results (Table 1). CoreHD (Zdeborová et al., 2016), reverse dismantling (Duan et al., 2026) (dynamic variant), CND and LGD-NA (Adler et al., 2026) are run with their published default configurations, NIRM is trained on tiny synthetic networks with its published protocol (Zhang & Wang, 2022); Structural-AR is run with the larger of its default configuration and a random search of 20 configurations selected on validation cells, and is trained with 5 seeds at the same training wall-clock as ELDOR. Equal-time comparisons use independent restarts of LGD-NA and MIND with random tie-breaking (Table 16). Selected hyperparameters and compute are listed in Table 9.

Metrics and statistics. We report the dismantling AUC of Eq. (1) at $\rho = 0.1$, normalized per network so that MIND $= 100$, wall-clock time, peak memory and NFE. Uncertainty has two sources: independently trained models (5 seeds) and decoding randomness (3 decoding seeds for sampled decoding). Tables report the standard deviation across trained models after averaging over decoding seeds; paired comparisons use networks as the paired units (Table 22).

K.1 Evaluation Metric

Table 8: Training schedules and exact-state sampling, fixed before any evaluation.

Quantity	Schedule
ϵ_μ (behavior mixing)	linear $0.5 \to 0.05$ over the first 40% of steps, then constant
κ (EMA rate of $\bar{\theta}$)	cosine $0.99 \to 0.999$ over all steps
λ_{ex}	1.0 for 10^4 steps, then exponential decay (half-life $2 \cdot 10^4$) to $\lambda_{\text{ex}}^{\min}$=0.1
$\bar{\beta}$	geometric $0.5 \to 4.0$ over the first 60% of steps
$n_{\max}$	$20 \to 1{,}000$, doubled every $1.5 \cdot 10^4$ steps
exact-state sampling	per exact graph: 50% states on optimal trajectories, 30% on $q^\star$ samples, 20% uniform over reachable states
optimizer	Adam, lr $3 \cdot 10^{-4}$ with cosine decay, $2 \cdot 10^5$ steps, batch of 256 states

Table 9: Hyperparameter search spaces and selected values, chosen on validation cells only.

Hyperparameter	Search space	Selected
L	$\{2, 4, 6, 8, 12\}$	6
d	$\{64, 128, 256\}$	128
α	$\{0, 0.5, 1\}$	0.5
$\bar{\beta}$ (final)	$\{1, 4, 16\}$	4
λ_{TB}	$\{0, 0.1, 1\}$	0.1
n_{ex}	fixed	20
$n_{\max}$	$\{500, 1000, 2000\}$	1000
learning rate	log-unif. $[10^{-4}, 10^{-3}]$	$3 \cdot 10^{-4}$
T (units of $\bar{\beta}$)	$\{0.5, 0.75, 1, 1.5\}$	1
K	$\{1, 4, 16, 64\}$	all reported
$k_{\max}$	$\{1, 16, 64, 256\}$	16
w, I (refinement)	$\{5, 20\} \times \{10^2, 10^3\}$	$20, 10^3$

The implementation computes, for every method and network, $\mathcal{R}(\pi; G) = n^{-1} \sum_{t=0}^{B_\rho - 1} |C_{\max}(S_t)|/n$ with B_ρ the first step at which $|C_{\max}(S_t)| \le \rho n$; the initial state has weight one, the collapse state and all later states weight zero, and no interpolation is used. This equals $n^{-1}(c(V) + \Phi(\pi))$ with the running cost of Eq. (7), so the reported metric and the Feynman–Kac objective differ only by the constant $c(V)/n$, including threshold handling. Values are normalized per network by MIND evaluated with the same formula from its released checkpoint. The nine quoted methods were reported with Simpson's rule; for the eight of them with public code we recomputed the formula above from their released dismantling orders, which changes the normalized values by at most 0.4 on any network and leaves every domain mean within 0.2 and every average rank unchanged.

K.2 Table Details

This subsection gives the per-table settings that the captions omit. Table 1 and Table 14 cover all 48 real networks, with statistics measured from the released files (Table 11); domain columns are unweighted means over networks, excluding the two `n/a` entries of the source. Table 12 averages 10 realizations per cell. In Table 3, every variant is retrained with matched compute and decoded greedily without refinement, so the full model equals greedy ELDOR and the Structural-AR row equals Structural-AR in Tables 1 and 12. In Table 16, $\pm$ for K=1 is the SD over 16 decoding seeds averaged over models, LGD-NA and MIND are deterministic given the graph, and the equal-time budgets are the per-network times of greedy and best-of-16 ELDOR. In Table 19, $k_{\max} = 1$ is sequential recomputation (Section 4) and $k_{\max} > 1$ is frozen-logit batching. In Table 20, total time is the wall-clock time of the whole method, and multi-start rows run the training-free method 16 times with random tie-breaking and keep the best trajectory.

Table 10: Compute and hardware used for training and evaluation.

Item	Value
GPUs	8× A100-80GB
CPUs	2× EPYC 7543, 32 cores
Memory	512 GB RAM
Deep learning	PyTorch 2.3, PyG 2.5
Graph library	NetworkX 3.3
ELDOR training, one seed	38 GPU-h
Learned models, 5 seeds each	1,420 GPU-h
Exact DP tables ($n \le 20$)	640 CPU-h
Evaluation, GPU	910 GPU-h
Evaluation, CPU	3,150 CPU-h
Hyperparameter search	1,180 GPU-h
Discarded runs	260 GPU-h

L Test Networks, Synthetic Protocol and Baselines

This appendix lists the real test networks, the synthetic test protocol and all baselines.

Real networks. Table 11 lists the 48 networks, 9 biological, 14 information, 16 social and 9 technological, with 128 to 1,402,673 nodes. Statistics are measured from the released files rather than copied from the original report. Networks with several components are kept whole; $c(S)$ is always normalized by the full n.

Table 11: All 48 real test networks, measured from the released files. All files are undirected and unweighted and free of multi-edges; the only preprocessing is the removal of self-loops. m raw: as stored; m proc.: after removing self-loops; comp.: connected components (all kept); LCC: largest component; iso.: isolated nodes; r: degree assortativity and Q: modularity, as published with the network files.

Dom.	Network	n	m raw	loops	m proc.	comp.	LCC	iso.	r	Q
Bio	arenas-meta	453	2,040	15	2,025	1	453		−0.214	0.445
Bio	dimacs10-celegansneural	297	2,148		2,148	1	297		−0.163	0.384
Bio	foodweb-baydry	128	2,106		2,106	1	128		−0.104	0.178
Bio	foodweb-baywet	128	2,075		2,075	1	128		−0.112	0.180
Bio	maayan-Stelzl	1,706	3,191	36	3,155	43	1,615	4	−0.187	0.618
Bio	maayan-figeys	2,239	6,432		6,432	9	2,217		−0.331	0.465
Bio	maayan-foodweb	183	2,452	18	2,434	1	183		−0.254	0.364
Bio	maayan-vidal	3,133	6,726	577	6,149	210	2,783	110	−0.097	0.678
Bio	moreno-propro	1,870	2,277	74	2,203	173	1,458	24	−0.152	0.847
Info	cit-HepPh	34,546	420,921	44	420,877	61	34,401		−0.006	0.725
Info	citeseer	384,413	1,743,018	6,873	1,736,145	6,751	365,154	359	−0.061	0.800
Info	com-amazon	334,863	925,872		925,872	1	334,863		−0.059	0.926
Info	com-dblp	317,080	1,049,866		1,049,866	1	317,080		+0.267	0.820
Info	dblp-cite	12,591	49,635	15	49,620	40	12,495		−0.046	0.633
Info	dimacs10-polblogs	1,224	16,715		16,715	2	1,222		−0.221	0.427
Info	econ-wm1	260	2,554	165	2,389	3	258	2	+0.032	0.268
Info	linux	30,837	213,747	530	213,217	12	30,817	3	−0.174	0.480
Info	p2p-Gnutella06	8,717	31,525		31,525	1	8,717		+0.052	0.388
Info	p2p-Gnutella31	62,586	147,892		147,892	12	62,561		−0.093	0.502
Info	subelj-jdk-jdk	6,434	53,658		53,658	1	6,434		−0.223	0.494
Info	subelj-jung-j-jung-j	6,120	50,290		50,290	1	6,120		−0.233	0.471
Info	web-EPA	4,271	8,909		8,909	8	4,253		−0.303	0.647
Info	web-Stanford	281,903	1,992,636		1,992,636	365	255,265		−0.112	0.927
Social	advogato	6,539	43,277	3,992	39,285	1,441	5,042	1,384	−0.061	0.461
Social	douban	154,908	327,162		327,162	1	154,908		−0.180	0.598
Social	ego-twitter	23,370	32,831		32,831	72	22,322		−0.478	0.895
Social	hyves	1,402,673	2,777,419		2,777,419	1	1,402,673		−0.023	0.771
Social	librec-ciaodvd-trust	4,658	33,116		33,116	39	4,562		+0.104	0.434
Social	librec-filmtrust-trust	874	1,309		1,309	95	610		+0.078	0.754
Social	loc-brightkite	58,228	214,078		214,078	547	56,739		+0.011	0.679
Social	loc-gowalla	196,591	950,327		950,327	1	196,591		−0.029	0.713
Social	munmun-digg-reply-LCC	29,652	84,781		84,781	1	29,652		+0.003	0.406
Social	munmun-twitter-social	465,017	833,540		833,540	1	465,017		−0.878	0.649
Social	opsahl-ucsocial	1,899	13,838		13,838	4	1,893		−0.188	0.262
Social	pajek-erdos	6,927	11,850		11,850	1	6,927		−0.116	0.696
Social	petster-hamster	2,426	16,631		16,631	148	2,000		+0.047	0.549
Social	slashdot-threads	51,083	117,378	805	116,573	1	51,083		−0.034	0.483
Social	slashdot-zoo	79,116	467,731		467,731	1	79,116		−0.075	0.341
Social	soc-Epinions1	75,879	405,740		405,740	2	75,877		−0.041	0.443
Tech	eu-powergrid	1,467	1,817		1,817	66	1,093		−0.064	0.926
Tech	gridkit-eupowergrid	13,844	17,277		17,277	59	13,478		+0.014	0.966
Tech	gridkit-north-america	16,167	20,242		20,242	34	14,990		+0.050	0.968
Tech	internet-topology	34,761	107,720		107,720	1	34,761		−0.215	0.610
Tech	london-transport-multi	369	430		430	1	369		+0.137	0.829
Tech	opsahl-openflights	2,939	15,677		15,677	11	2,905		+0.051	0.635
Tech	roads-california	21,048	21,693		21,693	1	21,048		−0.002	0.975
Tech	roads-sanfrancisco	174,956	221,802		221,802	1	174,956		+0.083	0.986
Tech	tech-RL-caida	190,914	607,610		607,610	1	190,914		+0.025	0.856

Synthetic test protocol. The protocol evaluates on (i) Erdős–Rényi graphs with mean degree $\langle d \rangle = 4$, (ii) configuration-model graphs with $\langle d \rangle = 4$ and degree distribution $P(d) \sim d^{-2.5}$, and (iii) stochastic block models with groups of 100 nodes, $p_{\text{intra}} = 0.1$ and $p_{\text{inter}} = 5/|V|$, each at $n \in \{10^3, 10^4, 10^5\}$ with 10 realizations, i.e. 90 graphs (Table 12). We reproduce this protocol exactly; the configuration model and the 10^5-node sizes lie outside the training families and sizes of ELDOR.

Baselines. Table 13 lists the baselines: nine published methods (adaptive degree, betweenness, PageRank, min-sum, explosive immunization, GND, FINDER, GDM, MIND), whose reported values we quote (Table 14), and the baselines added in this paper (CoreHD, reverse dismantling, CND, LGD-NA, NIRM, Structural-AR). All are compared with the dismantling AUC of Eq. (1) ($\rho = 0.1$; Appendix K.1), normalized per network so that MIND $= 100$. The added baselines and ELDOR are run under the same protocol, so that every column of Tables 12 and 14 is comparable. Every method runs on one GPU with a limit of 24 h per network; a run that exceeds it is reported as out of time.

Table 12: Synthetic test protocol, all methods and cells: dismantling AUC ($\rho = 0.1$) normalized so that MIND $= 100$ ($\downarrow$), mean over 10 realizations per cell; abbreviations as in Table 14. The overall values of the first nine methods are published; their per-cell values, published only as plots, are re-measured here. For ELDOR, the CM rows are an unseen family and the 10^4 and 10^5 columns exceed its largest training graph; the last three rows aggregate the groups of Table 2. Shading marks the best (bold) and second-best value in each column.

Model	n	AD	BC	PR	MS	EI	GND	FINDER	GDM	MIND	NIRM	S-AR	CoreHD	CND	RD	LGD-NA	ELDOR g	ELDOR 16
ER	10^3	104.4	126.1	118.0	103.4	102.3	118.6	104.5	110.6	100.0	100.1	103.2	104.7	101.5	99.0	98.1	93.3	**91.3**
ER	10^4	105.8	123.3	117.1	103.3	104.4	116.6	106.7	110.0	100.0	102.2	103.5	103.4	101.0	97.9	98.8	95.7	**91.6**
ER	10^5	107.1	123.9	116.1	103.2	105.0	116.9	104.0	110.2	100.0	103.0	103.9	105.3	102.2	100.6	98.0	94.9	**91.5**
CM	10^3	107.5	123.4	117.8	105.3	103.2	121.6	107.0	110.9	100.0	101.8	102.0	103.1	101.7	98.8	99.1	96.0	**92.6**
CM	10^4	105.3	123.3	118.0	104.5	104.3	115.8	105.0	108.4	100.0	101.0	102.8	102.4	101.1	99.8	99.8	95.8	**92.0**
CM	10^5	106.1	124.3	119.1	105.3	109.6	117.0	103.0	108.9	100.0	102.4	103.3	104.4	101.4	100.1	97.0	96.8	**93.0**
SBM	10^3	108.1	124.4	117.3	100.5	104.2	117.3	103.5	106.2	100.0	101.7	101.8	103.7	100.2	99.2	97.4	94.4	**91.3**
SBM	10^4	106.0	125.7	120.6	101.6	105.0	117.4	102.2	108.5	100.0	100.2	101.8	102.6	99.7	99.1	97.3	92.8	**91.4**
SBM	10^5	106.6	126.6	121.4	101.6	107.4	118.3	106.8	108.8	100.0	101.2	100.8	102.4	98.5	97.0	96.4	94.4	**90.5**
Overall		106.3	124.6	118.4	103.2	105.0	117.7	104.7	109.2	100.0	101.5	102.6	103.6	100.8	99.1	98.0	94.9	**91.7**
Unseen family (CM)		106.3	123.7	118.3	105.0	105.7	118.1	105.0	109.4	100.0	101.7	102.7	103.3	101.4	99.6	98.6	96.2	**92.5**
Seen family, unseen size ($n \geq 10^4$)		106.4	124.9	118.8	102.4	105.4	117.3	104.9	109.4	100.0	101.6	102.5	103.4	100.3	98.7	97.6	94.5	**91.2**
Seen family and size ($n = 10^3$)		106.2	125.3	117.7	102.0	103.2	117.9	104.0	108.4	100.0	100.9	102.5	104.2	100.8	99.1	97.8	93.8	**91.3**

Table 13: Baseline methods with their family and published complexity for adjacency-list input ($\langle d^2 \rangle$: second moment of degree); for ELDOR see Proposition 12.

Method	Family	Complexity
Adaptive degree, dynamic betweenness, PageRank, CoreHD	centrality heuristics	—
Min-sum (BPD)	approximate theory	$O(\lvert V\rvert \log \lvert V\rvert) + O(\lvert E\rvert)$
Explosive immunization	approximate theory	$O(\lvert V\rvert \log \lvert V\rvert)$
GND	approximate theory	$O(\lvert V\rvert \log^{2+\epsilon} \lvert V\rvert)$
FINDER	learned (RL)	$O(\lvert V\rvert \log \lvert V\rvert + \lvert E\rvert)$
GDM	learned (supervised)	$O(\lvert V\rvert \langle d^2\rangle + \lvert E\rvert)$
MIND	learned (RL)	$O(\lvert V\rvert + \lvert E\rvert)$
NIRM	learned (ranking, tiny training graphs)	—
Structural-AR	learned (this paper's control)	—
Reverse dismantling (RD)	reinsertion order, dynamic	—
CND, LGD-NA	recomputed latent geometry	—
ELDOR	learned (Doob control)	Proposition 12

M Additional Experimental Results

This appendix reports six analyses that support or delimit the claims of Section 5. Each subsection first states what is measured and how, then what each part of the table or figure is designed to test, then what the results show, and finally what they do not show. All results are zero-shot. Each learned method is evaluated with one checkpoint selected on synthetic validation cells, with $\rho = 0.1$, the budget rule of Section 5.1 and sampling temperature $T = 1$ (Section 3.3). Our learned methods are trained 5 times independently; a reported value is the mean over these 5 models, each first averaged over 3 decoding seeds, and $\pm$ is the standard deviation across models (training-free methods: 5 tie-breaking seeds). Significance is assessed with Wilcoxon signed-rank tests over networks with Holm correction, and runtimes are wall-clock times per graph on the hardware of Table 10.

M.1 Real Networks: Structure, Decoding and Failure Cases

Table 15 regroups the 48 real networks of Table 1 into four quartiles of 12 networks by modularity Q, measured from the released files (Table 11). Table 16 breaks ELDOR down by decoder and compares it with MIND and LGD-NA restarted for the same per-graph time. Both tables are in the units of Table 1, so a lower value always means faster fragmentation relative to MIND; per-network values are in Table 14.

What each grouping tests. *Modularity quartiles.* Domain labels mix networks of very different structure, whereas modularity measures how strongly a network is held together by a few cuts between communities. In the most modular networks the giant component falls apart only when a whole cut is removed, which is the regime in which the one-step immediate term of Eq. (8) sees least

Table 14: Dismantling AUC ($\rho = 0.1$) on every real test network, normalized per network so that MIND $= 100$ (↓). The first nine columns are published values (n/a as published; source in Table 1); the last eight are run under the same protocol. AD adaptive degree, BC betweenness, PR PageRank, MS min-sum, EI explosive immunization, S-AR Structural-AR, ELDOR g greedy, ELDOR 16 best-of-16. †: the budget of ELDOR had to be raised above $0.5n$ to reach collapse, outside its training range. Shading marks the best (bold) and second-best value in each column.

Network	AD	BC	PR	MS	EI	GND	FINDER	GDM	MIND	NIRM	S-AR	CoreHD	CND	RD	LGD-NA	ELDOR g	ELDOR 16
arenas-meta	100.6	138.3	111.5	117.0	117.0	125.3	107.3	97.1	100.0	103.5	102.7	103.4	102.8	99.9	99.4	97.6	**94.0**
dimacs10-celegansneural	117.3	142.5	132.5	133.8	120.3	96.7	115.5	113.8	100.0	101.0	105.1	113.6	102.4	97.6	97.5	94.3	**91.3**
foodweb-baydry†	108.4	120.9	133.2	115.4	113.8	115.5	107.7	110.6	100.0	102.7	102.4	105.9	97.4	97.3	**95.4**	100.6	97.8
foodweb-baywet†	107.4	121.7	133.3	115.2	112.0	118.9	106.4	107.8	100.0	104.0	104.8	104.9	98.2	99.6	97.1	94.7	**91.5**
maayan-Stelzl	104.9	140.3	116.1	114.3	116.9	147.6	103.2	102.4	100.0	102.8	106.0	104.0	102.2	99.2	100.0	97.6	**93.8**
maayan-figeys	103.7	157.9	131.0	133.2	160.5	107.4	102.2	103.0	100.0	102.9	108.5	100.3	95.8	96.5	96.3	94.2	**91.7**
maayan-foodweb†	120.2	132.2	124.5	162.0	154.9	116.8	104.5	104.8	100.0	104.6	105.2	116.7	95.9	94.9	93.5	90.9	**87.9**
maayan-vidal	107.2	128.6	113.2	118.4	105.4	115.2	101.0	103.7	100.0	101.5	106.3	103.3	99.7	95.5	95.7	93.2	**90.1**
moreno-propro	124.1	157.6	135.5	156.0	94.4	124.4	114.6	107.4	100.0	101.6	102.9	121.4	99.7	98.5	97.7	95.6	**93.3**
cit-HepPh†	121.5	138.3	133.7	125.3	115.6	**90.5**	120.1	121.0	100.0	104.0	103.1	116.0	95.8	97.4	97.2	94.9	93.0
citeseer	109.3	165.7	128.0	118.9	106.0	109.2	107.0	106.9	100.0	101.3	104.5	105.6	101.1	97.6	98.0	96.9	**93.5**
com-amazon	123.1	172.1	122.6	152.8	**75.5**	n/a	122.2	107.0	100.0	100.6	104.6	120.0	96.0	96.6	93.0	90.8	87.8
com-dblp	114.3	121.0	109.5	180.0	**88.5**	106.9	115.6	97.5	100.0	103.4	106.7	109.0	98.9	94.2	94.1	92.6	89.7
dblp-cite	124.8	130.8	126.1	153.7	122.8	124.5	117.0	110.0	100.0	103.9	105.6	122.3	101.5	97.5	**97.0**	101.2	97.3
dimacs10-polblogs†	111.5	125.4	122.9	117.5	126.2	117.5	107.8	109.3	100.0	104.6	105.5	106.2	97.1	97.2	**96.1**	100.1	97.6
econ-wm1	103.4	125.0	123.4	101.5	120.6	122.8	92.4	94.2	100.0	105.0	103.4	100.9	97.2	95.2	94.1	91.8	**89.0**
linux	128.6	404.0	194.7	166.0	93.5	108.3	124.2	110.7	100.0	103.0	103.8	123.9	99.4	98.8	97.5	94.3	**91.2**
p2p-Gnutella06	115.5	126.0	118.8	115.8	115.5	136.8	112.2	106.4	100.0	103.5	104.3	112.3	102.0	100.3	99.4	97.5	**94.7**
p2p-Gnutella31	110.7	131.5	112.0	110.8	114.3	135.6	107.4	101.5	100.0	103.2	102.8	107.2	98.8	97.0	95.2	91.2	**88.7**
subelj-jdk-jdk	112.6	339.3	141.0	141.4	110.6	105.2	102.2	97.8	100.0	104.2	104.5	109.8	102.3	102.0	100.0	97.9	**94.8**
subelj-jung-j-jung-j	124.1	340.3	153.2	145.5	120.4	103.8	109.5	101.7	100.0	103.1	106.3	119.8	99.8	98.4	96.9	93.2	**90.5**
web-EPA	109.3	147.3	112.2	169.0	150.9	158.6	106.7	106.9	100.0	101.8	103.6	106.0	102.0	97.9	97.6	93.5	**89.7**
web-Stanford	209.8	239.9	173.3	n/a	**65.1**	114.8	207.2	98.9	100.0	105.2	102.6	204.4	99.3	98.2	96.1	93.5	90.9
advogato	107.6	121.1	158.5	119.0	117.8	113.9	103.9	105.6	100.0	106.4	102.9	104.3	101.1	94.8	95.3	93.9	**91.3**
douban	108.0	117.4	102.3	126.9	115.4	115.6	95.0	95.7	100.0	102.3	103.9	105.0	100.9	97.7	96.8	94.2	**91.1**
ego-twitter	110.3	135.0	109.1	169.4	104.3	118.3	112.2	101.2	100.0	100.7	106.5	107.0	98.5	95.3	94.2	91.1	**88.7**
hyves	111.5	159.6	104.2	135.3	169.1	110.7	102.6	100.5	100.0	104.1	101.1	109.3	99.8	95.8	95.8	95.0	**92.1**
librec-ciaodvd-trust	132.4	129.3	135.6	142.4	145.6	127.3	116.1	112.5	100.0	104.2	105.1	129.9	99.8	96.6	**95.8**	100.1	97.2
librec-filmtrust-trust	122.8	166.0	137.6	176.1	117.8	113.7	125.2	109.8	100.0	105.1	103.6	120.4	100.3	99.5	98.3	100.3	**97.8**
loc-brightkite	112.4	139.1	118.8	121.7	111.4	114.2	109.7	113.9	100.0	103.2	106.4	109.0	97.8	**94.8**	**94.8**	100.4	96.9
loc-gowalla	106.5	148.8	114.8	112.3	101.5	110.0	105.5	106.5	100.0	102.8	108.4	102.8	99.9	96.2	95.9	93.8	**91.4**
munmun-digg-reply-LCC	104.7	124.9	113.1	105.1	116.2	123.5	103.3	106.2	100.0	104.5	106.7	103.7	100.4	101.5	99.7	96.0	**92.2**
munmun-twitter-social	101.3	103.3	**90.7**	138.4	124.9	103.6	102.9	98.5	100.0	104.0	103.8	101.6	101.1	98.8	97.6	94.1	91.2
opsahl-ucsocial	107.7	118.4	114.2	117.3	127.8	131.6	105.0	107.8	100.0	102.3	102.7	103.6	105.0	98.9	99.2	97.0	**93.6**
pajek-erdos	103.8	107.1	103.7	123.3	120.3	112.7	102.5	100.4	100.0	104.6	103.7	101.4	98.7	98.1	96.1	93.2	**90.1**
petster-hamster	128.4	129.6	141.4	174.5	108.7	96.8	129.2	104.6	100.0	103.9	104.6	125.9	101.8	101.2	101.1	98.4	**95.6**
slashdot-threads	103.6	118.9	108.8	122.5	127.5	104.2	101.8	104.1	100.0	105.2	103.1	100.5	97.6	94.7	95.2	92.2	**89.0**
slashdot-zoo	103.8	134.6	112.1	121.9	131.2	107.6	101.9	108.4	100.0	102.5	106.5	103.3	102.3	99.5	99.3	95.7	**92.3**
soc-Epinions1	100.7	120.3	104.1	117.4	115.6	99.6	97.0	99.8	100.0	104.1	106.0	104.3	101.9	102.3	100.3	96.1	**92.8**
eu-powergrid	151.5	190.5	196.5	317.5	80.6	82.8	161.7	109.1	100.0	109.4	113.6	139.9	84.0	**73.7**	78.7	81.8	78.5
gridkit-eupowergrid	136.0	175.8	132.2	256.1	**36.4**	64.0	132.2	97.6	100.0	110.1	114.7	124.2	83.4	77.5	80.0	82.4	80.1
gridkit-north-america	124.7	149.4	133.5	184.1	**26.3**	56.4	137.0	95.0	100.0	113.0	113.4	114.3	82.5	72.4	77.9	80.8	77.7
internet-topology	102.0	126.5	112.4	142.6	117.2	98.4	102.2	102.9	100.0	108.7	111.7	95.9	81.7	**75.9**	79.5	83.5	79.6
london-transport-multi	129.2	152.9	134.1	124.3	112.0	93.5	129.0	105.4	100.0	107.8	110.4	119.5	85.1	**74.7**	81.2	82.2	80.7
opsahl-openflights	135.2	131.3	130.9	167.1	116.8	107.6	120.5	106.2	100.0	109.4	112.1	125.0	86.5	**76.2**	81.3	84.8	80.8
roads-california	192.8	882.0	125.1	114.9	**23.4**	78.0	116.3	92.2	100.0	109.6	113.3	177.2	88.3	77.3	82.8	87.5	84.7
roads-sanfrancisco	184.5	51.6	258.8	160.4	**15.5**	30.0	176.4	92.0	100.0	113.6	110.8	169.6	80.1	76.0	80.3	85.0	82.2
tech-RL-caida	120.7	172.6	121.3	157.3	**78.5**	112.1	116.2	106.9	100.0	109.8	110.1	111.8	84.8	80.1	83.2	86.4	82.6
Mean (all 48)	119.9	167.8	129.5	142.8	108.0	109.1	115.0	104.2	100.0	104.6	106.0	115.5	96.8	93.7	93.9	93.2	90.2
Avg. rank	12.6	15.7	14.1	15.3	10.8	11.2	10.4	8.8	6.2	9.0	9.8	10.6	5.8	3.9	3.5	3.3	1.8

and the learned future term must do most of the work. *Decoders.* Greedy decoding is the policy itself; a single sampled trajectory ($K{=}1$) is one draw from the same kernel; best-of-16 adds search; batched greedy is the engineering approximation of Section 3.3. *Equal time.* Restarting a baseline for the time ELDOR uses tests whether the gain of ELDOR is only a matter of compute.

Reading the tables. Best-of-16 ELDOR is best in three of the four quartiles and in every domain except the technological one, and the exceptions are largely the same networks: 7 of the 9 technological networks have $Q \geq 0.82$ and fall in the most modular quartile.

Modular networks favor separator-based methods. In Q1, explosive immunization reaches 66.7, reverse dismantling 84.5 and best-of-16 ELDOR 84.7, while greedy ELDOR (87.5) is behind LGD-NA (86.6) and adaptive degree rises to 143.4. These networks are dismantled by removing a few long

Table 15: Dismantling AUC on the 48 real networks grouped by modularity quartile (12 networks each; Q1 most modular), normalized so that MIND = 100 (↓). The upper block uses published values; the other blocks are run under the same protocol. Shading marks the best (bold) and second-best value in each column.

Method	Q1 0.82–0.99	Q2 0.63–0.80	Q3 0.45–0.62	Q4 0.18–0.44
Adaptive degree	143.4	113.8	111.2	111.1
Betweenness	216.7	138.8	188.8	126.8
PageRank	146.0	117.8	131.9	122.3
Min-sum	179.3	138.3	134.5	122.1
Explosive immunization	**66.7**	121.9	118.3	125.0
GND	89.2	114.2	113.5	117.9
FINDER	136.7	110.1	107.3	105.8
GDM	100.9	107.0	102.3	106.8
MIND	100.0	100.0	100.0	100.0
NIRM	107.1	103.8	104.1	103.6
Structural-AR	109.1	105.2	105.1	104.8
CoreHD	134.9	110.2	108.3	108.8
CND	90.0	98.7	98.7	100.0
Reverse dismantling (RD)	84.5	95.4	96.3	98.4
LGD-NA	86.6	95.4	96.1	97.3
ELDOR greedy	87.5	95.1	94.0	96.2
ELDOR best-of-16	84.7	**92.0**	**90.9**	**93.2**

community cuts, and a method that scores nodes through a one-step afterstate cost must discover each cut node by node.

Elsewhere the learned future term pays off. From Q2 to Q4, best-of-16 ELDOR is best (92.0, 90.9, 93.2) and greedy ELDOR second, ahead of LGD-NA by 0.3, 2.1 and 1.1 points. These are the networks in which the value of a removal depends on what the remaining budget can still achieve, the quantity that r_θ is trained to estimate.

The pattern is specific to the exact–learned split. MIND, NIRM and Structural-AR vary by at most 4.3 points across quartiles and never lead, so the quartile pattern is not a property of learned dismantlers in general.

Table 16: ELDOR on the 48 real networks under sequential greedy ($k_{\max}$=1), batched greedy ($k_{\max}$=16), single-sample and best-of-16 decoding, dismantling AUC normalized so that MIND = 100 (↓) and averaged within each domain; ±: SD over trained models and decoding seeds. MIND and LGD-NA are shown for reference; in the last two rows they are restarted for the per-graph time of batched greedy and of best-of-16 ELDOR.

Domain	Networks	MIND	LGD-NA	greedy	batched	K=1	best-of-16
Biological	9	100.0	97.0	95.4±1.0	95.8	96.9±1.6	92.4±0.6
Information	14	100.0	96.6	95.0±1.0	95.3	96.5±1.6	92.0±0.6
Social	16	100.0	97.2	95.7±1.0	96.1	97.2±1.6	92.7±0.6
Technological	9	100.0	80.5	83.8±1.0	84.2	85.3±1.6	80.8±0.6
All	48	100.0	93.9	93.2	93.6	94.7	90.2
Equal time, batched-greedy budget	48	99.2	93.7	–	93.6	–	–
Equal time, best-of-16 budget	48	97.9	92.2	–	–	–	90.2

Sampling helps only through selection. Greedy decoding (93.2) is better than a single sampled trajectory (94.7), whose variability over seeds is also larger (±1.6 against ±1.0); only selecting the best of 16 samples turns sampling into a gain (90.2). Batching costs 0.4 points on average (Appendix M.4).

At equal time, the greedy policy is not ahead. Restarted for the per-graph time of batched greedy ELDOR, LGD-NA reaches 93.7 against 93.6, a tie; restarted for the time of best-of-16, it reaches 92.2 against 90.2. MIND gains little from restarts (99.2 and 97.9). The advantage of ELDOR at equal compute therefore comes from search over its own kernel, whose samples are diverse enough for selection to help, rather than from a better single trajectory.

Where ELDOR ties or loses. On the technological domain explosive immunization remains best (67.4 on average); reverse dismantling is ahead of best-of-16 ELDOR on all nine technological

networks, and LGD-NA on four of them, including both road networks. There the separators are long, globally ambiguous cuts beyond the effective receptive field of message passing, and a single articulation step does not reveal them; on roads-sanfrancisco and roads-california explosive immunization reaches 15.5 and 23.4. On 5 of the 48 networks, all dense (mean degree above 19, marked † in Table 14), there is little articulation information to exploit, the budget of ELDOR must be raised above $0.5n$, outside its training range, and all methods lie close together.

What these results do not show. Modularity is a proxy for separator structure, not a measurement of it, and each quartile contains only 12 networks, so we do not test the quartile differences for significance. The failure cases are the complement of our claim: the formulation helps where the value of a removal depends on the remaining budget and on coordination among removals, not where one structural statistic already identifies the cut.

M.2 Potential Error and Consistency

Table 17 checks the premise of Proposition 8, a uniform bound on the error of the learned potential, on graphs where it can be checked, and reports two consistency diagnostics on the test sets, where it cannot. The same checkpoint as in every other table is used; no model is trained for this analysis.

What each block tests. *Exact graphs.* On 200 held-out graphs per size with $n \le 20$, the dynamic program of Appendix G gives $\log h$ exactly, so the error $\hat{\epsilon}$ of r_θ can be measured on every reachable state, weighted by how often the exact kernel visits it, and the measured path divergence can be compared with the bound $2\hat{\epsilon}_{\max}B$. *Test sets.* On the real networks and the synthetic protocol $\log h$ is unknown. We report two conditions that the exact potential satisfies: a zero one-step residual (Eq. (11)) along greedy trajectories, and a perfect rank agreement between u_θ and rollout values of candidate successors.

Table 17: Potential error and consistency diagnostics. **Top:** the premise of Proposition 8 on held-out exactly solvable graphs (200 per size from the training families, test seeds, $B = 0.4n$). Errors $\hat{\epsilon} = |r_\theta(v; S, b) - \log h_{b-1}(S \setminus \{v\})|$ are taken over every candidate of every reachable state, weighted by the probability that $q^\star$ visits the state; we report their mean, 99th percentile and maximum, the measured $D_{\mathrm{KL}}(\mathbb{P}_\theta \| \mathbb{P}^\star)$ and the bound $2\hat{\epsilon}_{\max}B$. **Bottom:** the test sets, where $\log h$ is unknown and the premise cannot be checked: normalized one-step residual $|\delta_t|/s_t$ of Eq. (11) along greedy trajectories, and rank correlation ρ_s^{roll} between u_θ and 64-sample rollout values.

Exact targets	B	$\rho_s(u_\theta, \log h) \uparrow$	mean $\hat{\epsilon}$	$q_{0.99}(\hat{\epsilon})$	$\hat{\epsilon}_{\max}$	path KL$\downarrow$	bound $2\hat{\epsilon}_{\max}B$
$n = 8$	3	0.974	0.018	0.071	0.11	0.06	0.66
$n = 12$	5	0.961	0.029	0.12	0.19	0.19	1.90
$n = 16$	6	0.941	0.043	0.17	0.27	0.37	3.24
$n = 20$	8	0.928	0.052	0.21	0.34	0.52	5.44

Test sets (no exact target)	networks	residual$\downarrow$	$\rho_s^{\mathrm{roll}} \uparrow$
Real, biological	9	0.036	0.93
Real, information	14	0.038	0.92
Real, social	16	0.037	0.93
Real, technological	9	0.052	0.86
Real, all	48	0.040	0.91
Synthetic protocol	90	0.034	0.94

Reading the table. The premise holds with a clear margin on every exact size, and the test-set diagnostics are of the same order as on the exact graphs everywhere except on the technological networks.

The error grows slowly with size. The maximum error rises from 0.11 to 0.34 nats between $n = 8$ and $n = 20$, and the rank correlation with $\log h$ falls only from 0.974 to 0.928. Since exact labels are used only up to $n_{\mathrm{ex}} = 20$, this growth is the relevant warning sign for larger graphs, and it is moderate.

The bound is loose because large errors are rare. The measured path divergence stays about ten times below the bound (0.52 against 5.44 at $n = 20$). The bound charges every step with the worst error, whereas the mean error is about a sixth of the maximum and the exact kernel rarely visits the states on which the error is largest.

The test sets behave like the exact graphs, except on the technological networks. The residual (0.034 to 0.040) and the rollout agreement (0.91 to 0.94) match the exact-graph regime on the biological, information and social networks and on the synthetic protocol. On the technological networks the residual rises to 0.052 and the agreement falls to 0.86, the domain on which ELDOR also loses (Appendix M.1), so a less consistent potential and weaker dismantling occur together.

What the table does not show. The premise of Proposition 8 is established only for $n \leq 20$. The test-set diagnostics are necessary for it but not sufficient, so the KL guarantee is not claimed on the test sets.

M.3 Controlled Mechanism Holdouts

Table 18 evaluates held-out members of the fragmentation family (Appendix H) whose separators are built to defeat one-step or degree-based reasoning ($n \approx 10^3$, 60 graphs per column, $\mathcal{R} \times 100$, not normalized). No method is retrained, and the roles of nodes are never given to any method.

What each column and row tests. *Width* $w = 3, 4$. Connectors must be removed together, and training uses only $w \leq 2$; removing any single connector leaves the giant component unchanged, so the immediate term is flat and only a future term can identify the separator. *Symmetric.* Connectors are 1-WL-equivalent to ordinary members under the exact inputs, the case in which no structure-only network can separate them (Appendix I). *Non-articulation cuts.* Separators of 2–3 nodes, none of which is ever an articulation point, are hidden among decoy hubs. Among the rows, the myopic kernel ($r_\theta \equiv 0$) keeps the exact term and drops the learned one, and Structural-AR keeps the encoder and drops the Doob potential, so each isolates one half of the split.

Table 18: Controlled holdouts of the fragmentation family ($n \approx 10^3$, 60 graphs per column), $\mathcal{R} \times 100$ (↓, not normalized). Width w: connectors that must be removed together (training uses $w \leq 2$); symmetric: connectors 1-WL-equivalent to ordinary members under the exact inputs; non-articulation cuts: separators of 2–3 nodes, none of which is ever an articulation point, hidden among decoy hubs. MIND is run from its released checkpoint. Shading marks the best (bold) and second-best value in each column.

Method	$w = 3$	$w = 4$	symmetric	non-articulation cuts
MIND	22.9	24.8	13.9	21.6
Structural-AR	21.9	23.7	13.8	20.8
Myopic exact greedy ($r_\theta \equiv 0$)	24.1	26.2	13.7	22.9
Adaptive degree	27.9	29.6	14.1	26.4
Dynamic betweenness	**15.2**	16.4	13.6	**15.8**
CND	21.3	23.8	13.5	19.7
Reverse dismantling	19.8	22.1	13.3	18.8
LGD-NA	20.4	22.9	13.4	19.1
ELDOR greedy	16.9	18.2	13.4	17.6
ELDOR best-of-16	15.3	**16.3**	**13.1**	16.1

Reading the table. Best-of-16 ELDOR is best or second-best in every column, and greedy ELDOR is behind only dynamic betweenness on the width and cut columns.

Wider separators reward the learned future term. The greedy margin over LGD-NA grows with width, from 3.5 points at $w = 3$ to 4.7 at $w = 4$. The myopic kernel falls behind every separator-based heuristic, and its gap to greedy ELDOR, 7.2 and 8.0 points, is the contribution of r_θ on exactly the graphs where the immediate term is flat.

Direct scoring does not recover it. Structural-AR, with the same encoder and training data, is 5.0 and 5.5 points behind greedy ELDOR, so the gain comes from the Doob potential and its exact targets rather than from the encoder.

Symmetric candidates are a tie by construction. All methods lie within 1 point, as expected when neither the exact inputs nor message passing can tell connectors from members.

Non-articulation cuts favor path-based scores. Dynamic betweenness, which sees such cuts as shortest-path bottlenecks, is best (15.8), and best-of-16 ELDOR comes within 0.3 points.

What the holdouts do not show. The family is synthetic and built to isolate one mechanism per column; it shows which mechanism each part of ELDOR exploits, not how often such separators occur in real networks.

M.4 Decoding, Compute and Scaling

Figure 3 and Tables 19 to 21 separate the quality of ELDOR from its cost. All timings use the hardware of Table 10; the batch and efficiency tables use the same checkpoints as Table 1.

What each comparison tests. *Samples versus batch size.* Figure 3 crosses the number of trajectories K with the batch size $k_{\max}$ at $n = 10^3$, to find where extra compute buys most quality. *Batching error.* Table 19 compares frozen-logit batching with sequential decoding, the kernel of Section 4, domain by domain, because batching is an approximation that the theory does not cover. *Cost.* Table 20 splits time into preprocessing and decoding and adds memory and neural function evaluations (NFE). *Scaling.* Table 21 reports time and peak memory from $n = 200$ to $5 \cdot 10^4$.

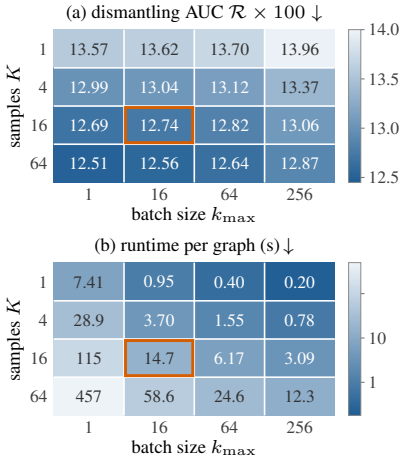

Figure 3: **Samples versus batch size** for ELDOR at $n = 10^3$. (a) Dismantling AUC ($\mathcal{R} \times 100$) and (b) runtime per graph for K sampled trajectories and batch size $k_{\max}$; darker is better in both panels, and the outlined cell is the configuration used throughout.

Reading the results. Quality is bought mainly with samples rather than with finer steps, and the batched approximation costs little quality for a large speed-up.

Samples matter more than finer steps. Quality improves mainly with K, whereas smaller batches cost far more time for a small gain: $k_{\max} = 1$ is dominated at equal runtime, and $k_{\max} = 16$ and 64 trade quality for time at nearly the same rate. We use $k_{\max} = 16$ with best-of-16 because it stays closer to sequential decoding.

Batching is a small, uniform error. At $k_{\max} = 16$ the mean changes by $+0.4\%$, between $+0.3$ and $+0.5$ in every domain, and no single network by more than 1.8 points; at $k_{\max} = 64$ the change grows to $+1.5\%$.

Table 19: Frozen-logit batching versus sequential greedy decoding ($k_{\max} = 1$, the kernel of Section 4) on the test sets, dismantling AUC normalized so that MIND $= 100$ ($\downarrow$). Δ: relative change of the mean; the last column gives the largest per-network change at $k_{\max} = 16$ and the network on which it occurs.

		greedy AUC$\downarrow$ at $k_{\max}$ =			Δ vs. $k_{\max}$=1 (%)		largest per-network
Test set	networks	1	16	64	16	64	change at $k_{\max}$=16
Real, biological	9	95.4	95.8	96.8	+0.4	+1.5	+1.1 (`foodweb-baydry`)
Real, information	14	95.0	95.3	96.4	+0.3	+1.5	+0.9 (`dimacs10-polblogs`)
Real, social	16	95.7	96.1	97.1	+0.4	+1.5	+1.0 (`hyves`)
Real, technological	9	83.8	84.2	85.1	+0.5	+1.5	+1.8 (`gridkit-eupowergrid`)
Real, all	48	93.2	93.6	94.6	+0.4	+1.5	+1.8
Synthetic protocol	90	94.9	95.3	96.3	+0.4	+1.5	+1.2 (SBM, n=10^5)

Search pays off up to about 16 samples. Batched greedy ELDOR takes 0.93 s per graph at $n = 10^3$ and sequential greedy 7.41 s; best-of-16 buys a clear gain at 14.7 s, best-of-64 shows diminishing returns at 58.6 s, and refinement is the most expensive variant (19.1 s).

Memory scales better than time. At $n = 5 \cdot 10^4$, greedy ELDOR takes 78.4 s per graph, against 44.3 s for LGD-NA and 1,680 s for Structural-AR, and needs 14.8 GB, less than every learned baseline. Best-of-16 multiplies time by about 15 but not memory, since samples are evaluated one after another.

Table 20: Runtime, memory and neural function evaluations (NFE) per graph at $n = 10^3$; the last column is $\mathcal{R} \times 100$ (↓) on synthetic validation graphs of that size. Multi-start rows keep the best of 16 tie-broken runs. Shading marks the best (bold) and second-best value in the last column.

Method	preproc. (s)	decode (s)	total (s)↓	CPU (GB)↓	GPU (GB)↓	NFE↓	traj.	$\mathcal{R} \times 100$ ↓
MIND	0.02	0.29	0.31	0.21	1.0	118	1	13.99
Structural-AR	0.31	1.46	1.77	0.26	1.3	300	1	14.16
CoreHD	0.00	0.02	0.02	0.08	–	0	1	15.30
CND	0.01	0.19	0.20	0.17	–	0	1	13.96
CND multi-start (16×)	0.01	3.19	3.20	0.17	–	0	16	13.72
LGD-NA	0.02	0.47	0.49	0.26	0.6	0	1	13.69
LGD-NA multi-start (16×)	0.02	7.82	7.84	0.26	0.6	0	16	13.44
ELDOR greedy, sequential	0.42	6.99	7.41	0.28	0.9	300	1	13.45
ELDOR greedy, batched	0.42	0.51	0.93	0.28	0.9	96	1	13.50
ELDOR stochastic (K=1)	0.42	0.53	0.95	0.28	0.9	98	1	13.62
ELDOR best-of-4	1.67	2.03	3.70	0.41	0.9	392	4	13.04
ELDOR best-of-16	6.61	8.09	14.70	0.72	0.9	1568	16	12.74
ELDOR best-of-64	26.40	32.20	58.60	1.63	0.9	6272	64	**12.56**
ELDOR best-of-16 + refine	6.61	12.44	19.05	0.94	0.9	1568	16	12.61

Table 21: Runtime per graph and peak memory across graph sizes, on the hardware of Table 10.

	time per graph (s)↓ at $n =$						peak memory (GB)↓		
Method	200	500	10^3	$5{\cdot}10^3$	10^4	$5{\cdot}10^4$	10^3	10^4	$5{\cdot}10^4$
MIND	0.08	0.16	0.31	1.52	3.08	16.20	1.00	7.52	34.60
Structural-AR	0.34	0.79	1.77	21.40	74.20	1,680	1.30	9.94	44.10
CoreHD	0.00	0.01	0.02	0.13	0.29	1.80	0.08	0.62	2.90
CND	0.03	0.09	0.20	1.41	3.22	21.60	0.17	1.48	7.10
LGD-NA	0.07	0.21	0.49	3.18	7.14	44.30	0.26	2.24	10.80
ELDOR greedy	0.21	0.46	0.93	5.32	12.10	78.40	0.90	3.58	14.80
ELDOR best-of-16	3.24	7.18	14.70	82.60	187	1,184	0.90	3.58	14.80

What the results do not show. ELDOR is not claimed to dominate runtime: training-free heuristics remain one to two orders of magnitude faster, and unbatched decoding is quadratic in n for $B = \Theta(n)$.

M.5 Statistical Tests

Table 22 gives the paired tests behind every significance statement of Section 5.2 and Appendix M.1. Networks are the paired units, and each value of ELDOR is the mean over 5 trained models.

How the tests are organized. The first block compares greedy ELDOR with each baseline over all 48 networks, which tests the policy itself. The second compares it with LGD-NA by domain and in its batched form, which tests where the overall tie comes from. The third compares best-of-16 with the strongest baselines and with LGD-NA restarted for equal time, which tests whether search is significant beyond compute.

Reading the table. Every significance claim of the paper is supported, and every tie is reported as a tie.

The greedy policy is tied with the strongest heuristics. Greedy ELDOR is significantly better than every learned baseline and than CND, and tied with reverse dismantling (-0.5, $p = 0.41$) and LGD-NA (-0.6, $p = 0.35$).

The tie hides a domain split. Against LGD-NA, greedy ELDOR is significantly better in each of the biological, information and social domains ($p \leq 0.004$) and significantly worse on the technological one ($+3.3$, $p = 0.008$); the two effects cancel over all 48 networks, and batching removes the overall difference entirely ($p = 0.72$).

Search is significant against every strong baseline. Best-of-16 is significantly better than MIND, reverse dismantling, LGD-NA and restarted LGD-NA at equal time (all $p < 10^{-4}$).

Table 22: Paired significance tests on the 48 real networks. Δ: mean paired difference in dismantling AUC (MIND $= 100$), negative favors ELDOR; $\pm$: half-width of the 95% confidence interval; p: Wilcoxon signed-rank test over networks with Holm correction within each block. Greedy denotes sequential greedy decoding.

Comparison	Networks	ΔAUC	p_{Holm}
ELDOR greedy vs. MIND	48	-6.8 ± 1.6	$< 10^{-4}$
ELDOR greedy vs. GDM	48	-11.0 ± 1.8	$< 10^{-4}$
ELDOR greedy vs. NIRM	48	-11.4 ± 2.6	$< 10^{-4}$
ELDOR greedy vs. Structural-AR	48	-12.8 ± 2.4	$< 10^{-4}$
ELDOR greedy vs. CND	48	-3.6 ± 1.0	$< 10^{-4}$
ELDOR greedy vs. RD	48	-0.5 ± 0.9	0.41
ELDOR greedy vs. LGD-NA	48	-0.6 ± 0.9	0.35
ELDOR greedy vs. LGD-NA, biological	9	-1.6 ± 0.8	0.004
ELDOR greedy vs. LGD-NA, information	14	-1.6 ± 0.6	$< 10^{-3}$
ELDOR greedy vs. LGD-NA, social	16	-1.5 ± 0.5	$< 10^{-3}$
ELDOR greedy vs. LGD-NA, bio. + inf. + soc.	39	-1.6 ± 0.4	$< 10^{-4}$
ELDOR greedy vs. LGD-NA, technological	9	$+3.3\pm1.4$	0.008
ELDOR batched greedy vs. LGD-NA	48	-0.3 ± 0.9	0.72
ELDOR best-of-16 vs. MIND	48	-9.8 ± 1.5	$< 10^{-4}$
ELDOR best-of-16 vs. RD	48	-3.5 ± 0.9	$< 10^{-4}$
ELDOR best-of-16 vs. LGD-NA	48	-3.7 ± 0.8	$< 10^{-4}$
ELDOR best-of-16 vs. LGD-NA restarts, equal time	48	-2.0 ± 0.6	$< 10^{-4}$

What the tests do not show. Holm correction is applied within each block, not across blocks, and the tests treat networks as independent units, although several come from the same source collections.