NeRD: Neural Replicator Dynamics for Maximum Clique Extraction

Anonymous authors
Paper under double-blind review

Abstract

Neural approaches to the Maximum Clique Problem (MCP) have made substantial progress, yet they share two persistent limitations: they depend on non-differentiable decoders to extract discrete solutions from soft node scores, and they offer no guarantee that the returned set is even a valid clique. We propose **NeRD** (*Neural Replicator Dynamics*), an end-to-end differentiable model that resolves both issues by combining a graph neural network with replicator dynamics on the probability simplex. The GNN maps an input graph to a distribution over nodes that concentrates mass where the clique is likely to be, providing a warm start that steers the dynamics away from suboptimal fixed points; the dynamics then refine this initialisation to an asymptotically stable fixed point whose support is, by construction, a maximal clique—no post-hoc decoder required. Training is driven by the Motzkin–Straus quadratic objective, which is differentiable through both components and yields a free lower bound on the clique number as a by-product. We further show that the GNN's learned representations act as a symmetry-breaking mechanism that resolves the historical Achilles' heel of replicator dynamics: its sensitivity to initialisation on hard, structured instances where the uniform starting point consistently falls into small local maxima. Experiments on eight real-world graph benchmarks and eight DIMACS-inspired synthetic families confirm that the two components are strictly complementary: NeRD achieves state-of-the-art clique-number prediction across the majority of benchmarks, consistently outperforming both GNN-only baselines and standalone replicator dynamics, with the largest gains precisely on the families where existing neural methods struggle most.

1 Introduction

The Maximum Clique Problem (MCP) asks for the largest fully connected subgraph of a given graph and is one of the canonical NP-hard problems on graphs Karp (1972); Bomze et al. (1999). It underlies maximum common subgraph computation in molecular biology and drug discovery, structural alignment in protein matching, cohesive-subgroup detection in social networks, and graph retrieval Malod-Dognin et al. (2010); Balasundaram et al. (2011); Raymond & Willett (2002); Cao et al. (2008). No polynomial-time algorithm can approximate the clique number $\omega(G)$ within a factor $n^{1-\varepsilon}$ unless $\mathrm{P} = \mathrm{NP}$ Hastad (1996); Zuckerman (2006), and exact ILP-based solvers do not scale beyond moderate instances, which has motivated decades of work on heuristics. Two largely separate threads have emerged. A classical line, rooted in the Motzkin–Straus theorem Motzkin & Straus (1965) and developed in the neural-network and game-theory communities of the nineties Pelillo (1999); Bomze et al. (1999); Pelillo & Torsello (2006), casts MCP as continuous quadratic optimisation on the simplex and solves it via replicator-style dynamics. A modern line, driven by graph neural networks (GNNs) Scarselli et al. (2009); Xu et al. (2018); Karalias & Loukas (2020); Min et al. (2022); Karalias et al. (2022); Sanokowski et al. (2023); Zhang et al. (2023); Sun & Yang (2023), maps graphs to soft node-level distributions that suggest where the clique is. The classical dynamics admit elegant convergence guarantees but are brittle to initialisation on hard instances; the neural pipelines learn from data but rely on non-differentiable decoders that offer no guarantee that the returned set is a clique at all. We argue, and demonstrate empirically, that each weakness is precisely the strength of the other approach, and that the two should be fused into a single end-to-end differentiable pipeline.

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.

The modern neural pipeline is, almost without exception, a two-stage process: a GNN produces soft node scores and a separate procedure—greedy peeling, beam search, conditional expectation, or random restarts—turns them into a discrete subset. EGN Karalias & Loukas (2020) pioneered this template with a probabilistic loss derived from the Erdős probabilistic method, and later work diversified the encoder with hybrid scattering networks Min et al. (2022), diffusion Sun & Yang (2023), GFlowNets Zhang et al. (2023), and autoregressive variational annealing Sanokowski et al. (2023); in all cases the decoder remains outside the computational graph, breaking gradient flow and providing no test-time feasibility certificate. MXNET Roy et al. (2025) achieves full differentiability, but recovers an estimate of $\omega(G)$ rather than the clique itself, at the cost of a Gumbel–Sinkhorn permutation search Mena et al. (2018). These limitations are most apparent on hard, structured DIMACS-style families such as `Brock`, `Keller`, or planted-clique `gen` Johnson & Trick (1996).

The classical thread offers exactly what the neural one lacks. The Motzkin–Straus theorem Motzkin & Straus (1965) reformulates MCP as $\max_{\mathbf{x}\in\Delta^n} \mathbf{x}^\top \mathbf{A}\mathbf{x}$, and Pelillo (1999) proved that the asymptotically stable fixed points of the discrete-time replicator dynamics on this programme are in one-to-one correspondence with the maximal cliques of G; Pelillo & Torsello (2006) extended the result to the payoff-monotonic family, including a faster exponential variant. Replicator dynamics (RD) is therefore a differentiable, highly parallelisable map whose convergence points are maximal cliques—no decoder, no post-hoc check. It has not displaced neural methods because, from the uninformative barycentre $\mathbf{x}_0 = \mathbf{1}/n$, it frequently converges to a small maximal clique on hard instances, which historically required expensive restarts or annealing schedules Bomze et al. (1999; 2000); Pelillo & Torsello (2006). What RD lacks is a principled starting point, and this is the role a GNN is suited to play: trained on the Motzkin–Straus objective, it produces a distribution $\mathbf{h} \in \Delta^n$ concentrated on nodes likely to belong to the maximum clique, which steers RD into the basins of large maximal cliques. The GNN supplies the symmetry-breaking initialisation RD lacks on highly symmetric instances such as Hamming or Keller graphs, and RD supplies the convergence certificate neural solvers lack. Training is end-to-end through both stages, and the quadratic functional at convergence yields a free lower bound $\omega(G) \geq 1/(1 - \mathbf{x}^{*\top}\mathbf{A}\mathbf{x}^*)$. We call the resulting model **NeRD** (*Neural Replicator Dynamics*).

We make three contributions. First, we introduce NeRD, a decoder-free, end-to-end differentiable MCP solver whose output is guaranteed by classical theory Pelillo (1999); Pelillo & Torsello (2006) to be the indicator vector of a maximal clique, with a free lower bound on $\omega(G)$. Second, we show that GNN-based initialisation addresses the historical weakness of replicator dynamics—sensitivity to the uniform starting point on symmetric or camouflaged instances—without random restarts or annealing schedules Pelillo & Torsello (2006), covering both linear and exponential payoff-monotonic variants. Third, on eight real-world TUDataset benchmarks and eight DIMACS-inspired synthetic families, NeRD attains the lowest clique-number MSE on the majority of benchmarks against eleven neural baselines, with the largest gains on the camouflaged `brock-style` and CSP-style `RB` families; ablations confirm what the classical literature predicts Bomze et al. (2000): the GNN alone and standalone RD both underperform, and the two components are strictly complementary.

2 Related Work

Neural solvers for combinatorial optimisation on graphs. The use of GNNs to tackle NP-hard graph problems has grown steadily since the early experiments with Hopfield-style networks Hopfield & Tank (1985); Bomze et al. (1999), and now spans supervised models trained on labelled solutions Li et al. (2018), unsupervised frameworks that learn directly from a differentiable surrogate loss Karalias & Loukas (2020); Karalias et al. (2022); Min et al. (2022); Tönshoff et al. (2019), reinforcement-learning policies that construct solutions sequentially Khalil et al. (2017); Kool et al. (2019); Sanokowski et al. (2023); Zhang et al. (2023), and diffusion-based solvers Sun & Yang (2023). Despite this methodological diversity, supervised approaches are bottlenecked by the cost of generating ground-truth solutions for hard instances Yehuda et al. (2020); Joshi et al. (2019), and reinforcement learning suffers from training instability and high sample complexity Nikishin et al. (2018). Unsupervised approaches sidestep both issues but inherit their own difficulties: continuous relaxations of discrete objectives often admit degenerate optima and require carefully designed correction terms Karalias & Loukas (2020), and—most relevant for our purposes—the soft node assignments they produce must be turned into a discrete subset by a separate decoder that lies outside

the computational graph, blocking end-to-end gradient flow and offering no guarantee that its output even satisfies the problem's feasibility constraints.

Neural methods for the maximum clique problem. On MCP specifically, EGN Karalias & Loukas (2020) derived a probabilistic loss from the Erdős probabilistic method and recovered a clique through sequential conditional-expectation decoding. Min et al. (2022) observed that low-pass GNNs oversmooth around the clique boundary and replaced the encoder with a hybrid scattering architecture combining band-pass wavelet filters with GCN-style filters; the loss and the greedy decoder remain those of EGN. Subsequent work has diversified the encoder further—DIFUSCO uses a graph-based diffusion model Sun & Yang (2023), GFlowNet-based solvers Zhang et al. (2023) construct solutions one node at a time, and VAG-CO Sanokowski et al. (2023) models MCP through an Ising formulation with an autoregressive policy and annealed entropy regularisation—but the decoder problem remains. MXNET Roy et al. (2025) takes a notably different route, reformulating MCP as locating the largest fully-dense square block in a permuted adjacency matrix searched by a Gumbel–Sinkhorn network Mena et al. (2018); this achieves full differentiability but at the cost of a combinatorially expensive permutation search, and it targets clique-*number* estimation rather than producing an actual clique. None of these methods offers a convergence certificate that the returned set is a valid maximal clique.

Replicator dynamics and the Motzkin–Straus connection. The continuous reformulation of MCP traces back to Motzkin and Straus Motzkin & Straus (1965), who showed that maximising $\mathbf{x}^\top \mathbf{A}\mathbf{x}$ over Δ^n is equivalent to finding the maximum clique. Pelillo (1999) established that the asymptotically stable fixed points of the discrete-time replicator equations applied to this programme are in exact one-to-one correspondence with the maximal cliques of G, with no post-hoc feasibility check required. Bomze and collaborators Bomze et al. (1999; 2000) studied the regularised variant $\mathbf{A} + \alpha\mathbf{I}$, which breaks ties among fixed points without altering the clique correspondence, and Pelillo & Torsello (2006) extended the result to the broader family of payoff-monotonic dynamics, showing that an exponential variant converges considerably faster while sharing the same fixed-point structure. The known weakness of all these dynamics is their sensitivity to initialisation: starting from the barycentre $\mathbf{x}_0 = \mathbf{1}/n$ on hard instances frequently leads to convergence at a small maximal clique Bomze et al. (2000), which historically motivated random restarts and perturbation schedules. Despite its theoretical elegance and natural differentiability, this thread has been almost entirely absent from the recent deep-learning literature on MCP. Our work is, to the best of our knowledge, the first to bring it into a modern end-to-end differentiable framework, and to show that the initialisation problem RD has historically suffered from is exactly the problem GNNs were built to solve.

3 PRELIMINARIES

Notation. Let $G = (V, E)$ be a simple undirected graph with $n = |V|$ vertices and adjacency matrix $\mathbf{A} \in \{0,1\}^{n\times n}$. A subset $C \subseteq V$ is a *clique* if every pair of nodes in C is connected, and is *maximal* if no superset of C is a clique; the *clique number* $\omega(G)$ is the size of the largest such set. We denote by $\Delta^n = \{\mathbf{x} \in \mathbb{R}^n_{\geq 0} : \sum_i x_i = 1\}$ the standard simplex, and by $\sigma(\mathbf{x}) = \{i : x_i > 0\}$ the support of $\mathbf{x} \in \Delta^n$. For a subset $S \subseteq V$ of size k, we write $\mathbf{x}^S \in \Delta^n$ for its barycentric indicator ($x^S_i = 1/k$ if $i \in S$ and 0 otherwise).

The Motzkin–Straus quadratic programme. The foundational result of Motzkin & Straus (1965) reformulates MCP as the standard quadratic programme

$$\max_{\mathbf{x}\in\Delta^n} \; f(\mathbf{x}) \;=\; \mathbf{x}^\top \mathbf{A}\,\mathbf{x}. \tag{1}$$

Its global optima are exactly the barycentric indicators $\mathbf{x}^{C^*}$ of the maximum cliques of G, with $f(\mathbf{x}^{C^*}) = 1 - 1/\omega(G)$. For any feasible $\mathbf{x} \in \Delta^n$ this yields the differentiable lower bound

$$\omega(G) \;\geq\; \frac{1}{1 - \mathbf{x}^\top \mathbf{A}\mathbf{x}}. \tag{2}$$

A subtlety is that equation 1 also admits spurious continuous optima outside the barycentres of cliques. Bomze regularisation Bomze et al. (1999) replaces $\mathbf{A}$ by $\mathbf{A}_\alpha = \mathbf{A} + \alpha\mathbf{I}$ with $\alpha \in (0, 1)$;

the regularised programme $\max_{\mathbf{x}\in\Delta^n} \mathbf{x}^\top \mathbf{A}_\alpha \mathbf{x}$ has the same maximisers but with a clean one-to-one correspondence between strict local maxima and *maximal* cliques of G. We adopt this regularisation throughout.

Discrete-time replicator dynamics. The discrete-time replicator map associated with the regularised programme, together with the exponential payoff-monotonic variant of Pelillo & Torsello (2006), reads

$$x_i^{(t+1)} \;=\; x_i^{(t)} \,\frac{(\mathbf{A}_\alpha \mathbf{x}^{(t)})_i}{\mathbf{x}^{(t)\top} \mathbf{A}_\alpha \mathbf{x}^{(t)}}, \qquad x_i^{(t+1)} \;=\; \frac{x_i^{(t)} \exp\!\big(\eta\,(\mathbf{A}_\alpha \mathbf{x}^{(t)})_i\big)}{\sum_j x_j^{(t)} \exp\!\big(\eta\,(\mathbf{A}_\alpha \mathbf{x}^{(t)})_j\big)}, \tag{3}$$

both of which keep $\mathbf{x}^{(t)}$ on the simplex at every step. Pelillo (1999) proved that f_α is a strict Lyapunov function of the linear map: it is non-decreasing along every trajectory and strictly increasing on non-stationary trajectories. Combined with Bomze regularisation, this implies that every asymptotically stable fixed point $\mathbf{x}^*$ is the barycentric indicator of a maximal clique $\sigma(\mathbf{x}^*)$ Pelillo (1999); Bomze et al. (1999), and the exponential variant shares the same fixed-point correspondence while converging substantially faster in practice.

Basins of attraction and the role of the starting point. Each maximal clique $C \subseteq V$ corresponds to an asymptotically stable equilibrium $\mathbf{x}^C$ of equation 3, and therefore to an open basin of attraction $\mathcal{B}(C) \subseteq \Delta^n$. The simplex thus decomposes into the disjoint union $\bigcup_C \mathcal{B}(C)$, modulo a measure-zero separatrix set. On hard graphs the basins of *small* maximal cliques cover substantial volume, and the barycentre $\mathbf{x}_0 = \mathbf{1}/n$ frequently lies inside one of them Bomze et al. (1999; 2000). This is the classical motivation for restarts and annealing, and it is the opening that NeRD exploits: the dynamics has the right asymptotic behaviour, but the choice of $\mathbf{x}^{(0)}$ determines which clique is recovered.

Problem statement. Given a set of training graphs $\mathcal{D} = \{(G_i, \omega(G_i))\}_{i=1}^N$ with known clique numbers, we seek an end-to-end differentiable model that, at inference, takes a graph G as input and returns $\mathbf{x}^* \in \Delta^n$ whose support is a maximal clique of G, together with a lower bound on $\omega(G)$. The model must require only differentiable components at training time.

4 METHODOLOGY

NeRD is a controlled coupling between a learned encoder and the classical replicator dynamics introduced in Section 3, illustrated end-to-end in Figure 1. As recalled there, every maximal clique of G corresponds to an asymptotically stable equilibrium of equation 3 on the Bomze-regularised payoff $\mathbf{A}_\alpha$, and the simplex decomposes into the basins of attraction of those equilibria. Standalone replicator dynamics has the right asymptotic behaviour but the wrong starting point: from $\mathbf{x}_0 = \mathbf{1}/n$ it routinely lands in the basin of a small maximal clique Pelillo & Torsello (2006); Bomze et al. (2000). NeRD replaces $\mathbf{x}_0$ with a learned distribution $g_\theta(G) \in \Delta^n$ produced by a graph neural network, and trains the GNN end-to-end through a short unrolled trajectory of equation 3, so that the seed it produces lies inside the basin of a large maximal clique. Section 4.1 describes the architecture; Section 4.2 explains why training through the dynamics with a small unrolling depth is the right design choice.

4.1 ENCODER AND REPLICATOR REFINEMENT

Structural features. Each node v_i enters the encoder with a seven-dimensional feature vector that depends only on local graph structure: the logarithm of its degree, its degree normalised by $|V| - 1$, its local clustering coefficient, its k-core number normalised by the maximum core, the density of the subgraph induced by its closed neighbourhood, the average degree of its neighbours normalised by $|V| - 1$, and the variance of its neighbours' degrees on the same scale. The choice is dictated by the geometry of the Motzkin–Straus landscape rather than by predictive convenience. Vertices in the interior of a large clique sit on dense, locally homogeneous neighbourhoods with low neighbour-degree variance; vertices on the boundary of a clique see a sharp drop in clustering and a spike in neighbour-degree variance. The first two coordinates carry scale, the next three carry local density,

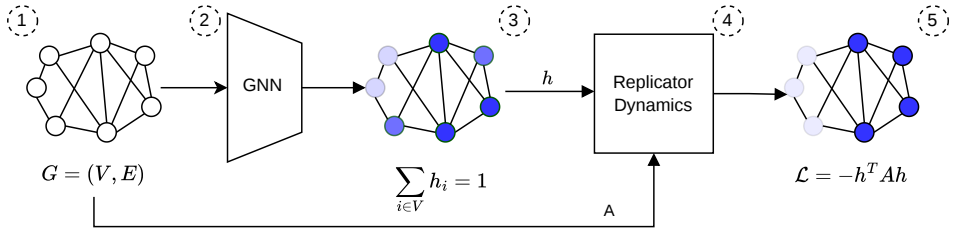


Figure 1: The NeRD pipeline. The input graph $G = (V, E)$ (1) is processed by a residual multi-scale GNN (2) whose softmax head produces a point $\mathbf{h} \in \Delta^n$ on the simplex (3). This point seeds the replicator dynamics equation 3 on the Bomze-regularised payoff $\mathbf{A}_\alpha = \mathbf{A} + \alpha\mathbf{I}$ (4), and the support of the converged fixed point (5) is the predicted maximal clique. The Motzkin–Straus loss $\mathcal{L} = -\mathbf{x}^\top \mathbf{A}_\alpha \mathbf{x}$ on the unrolled iterate is backpropagated through every unrolled replicator update.

and the last two encode the homogeneity that distinguishes interior from boundary. None of them identifies the maximum clique on its own; together they give the GNN enough leverage to perturb the symmetric region of the simplex from which equation 3 initialised at $\mathbf{1}/n$ cannot escape.

Residual GNN and projection to the simplex. The encoder g_θ stacks L residual message-passing layers of width d, $\mathbf{h}^{(\ell+1)} = \mathbf{h}^{(\ell)} + \phi^{(\ell)}(\mathbf{h}^{(\ell)}, \mathbf{A})$, where $\phi^{(\ell)}$ is a graph convolution chosen from `GCN`, `GAT`, `GraphSAGE`, or the scattering convolution of Min et al. (2022). Hidden representations from every layer are concatenated, $\mathbf{h}^{\text{cat}} = [\mathbf{h}^{(0)}, \ldots, \mathbf{h}^{(L)}]$, mirroring the multi-scale readout of Min et al. (2022) so that low-pass and band-pass features are jointly available to the head. A two-layer MLP maps $\mathbf{h}^{\text{cat}}$ to logits $\mathbf{z} \in \mathbb{R}^n$, which are projected onto the simplex by a softmax over nodes. The simplex, rather than the unit cube of independent Bernoulli probabilities used by Karalias & Loukas (2020), is the natural feasible set of the Motzkin–Straus programme equation 1 and of the replicator equations equation 3: encoder and dynamics share a single feasible set and a single objective, so no decoding step is needed to bridge the two domains.

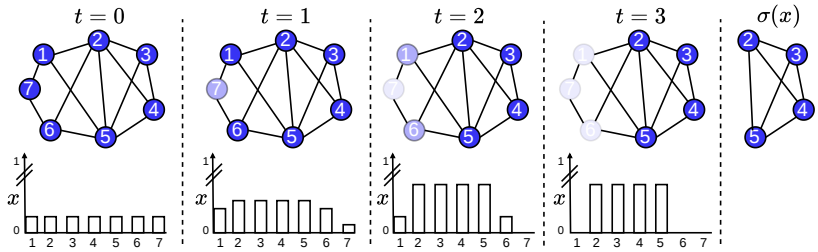


Figure 2: A representative trajectory of the replicator dynamics equation 3 on a seven-vertex graph whose maximum clique is $\{2, 3, 4, 5\}$. Each panel shows the simplex iterate $\mathbf{x}^{(t)}$ painted on the vertices, with intensity proportional to $x_i^{(t)}$, and the corresponding bar plot below. The leftmost panel ($t = 0$) is the uniform barycentre $\mathbf{x}^{(0)} = \mathbf{1}/n$; successive updates ($t = 1, 2, 3$) redistribute mass towards the maximum-clique vertices, and the rightmost panel $\sigma(\mathbf{x})$ shows the support of the converged fixed point as the recovered maximal clique.

Replicator refinement. Starting from $\mathbf{x}^{(0)} = g_\theta(G)$, NeRD unrolls T steps of either the linear update equation 3(left) or the exponential update equation 3(right). Both maps drive mass towards strategies whose payoff $(\mathbf{A}_\alpha \mathbf{x})_i$ exceeds the population mean $\mathbf{x}^\top \mathbf{A}_\alpha \mathbf{x}$, and both share the fixed-point correspondence with maximal cliques recalled in Section 3. We exploit this correspondence directly at inference: after T steps, or earlier once $\|\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\| < \tau$, the support of the iterate is read off as the predicted clique without any further decoding. The correspondence is asymptotic; at finite T we read the support at a threshold $\varepsilon = 10^{-5}$, and Appendix B (Table B.1) reports how often it is a valid, maximal clique. Figure 2 traces a representative trajectory from the barycentre:

successive replicator updates redistribute that mass onto a single clique characteristic vector while the remaining components are driven to zero.

4.2 Training and design rationale

Loss. Training is end-to-end with a Motzkin–Straus loss on the output of the unrolled dynamics,

$$\mathcal{L}(\mathbf{x}) \;=\; -\,\mathbf{x}^\top \mathbf{A}_\alpha\, \mathbf{x} \;+\; \lambda\big(\hat{\omega}(\mathbf{x}) - \omega(G)\big)^2, \qquad \hat{\omega}(\mathbf{x}) \;=\; \frac{1}{\sum_i x_i^2}. \tag{4}$$

The first term is the Motzkin–Straus quadratic equation 1: under Bomze regularisation, its strict local maximisers on Δ^n are exactly the clique characteristic vectors of G Pelillo (1999); Bomze et al. (1999). The second term is the inverse-participation-ratio estimator of Pelillo (1999): at the barycentric indicator $\mathbf{x}^C$ of a clique C one has $\sum_i (x_i^C)^2 = 1/|C|$, so $\hat{\omega}(\mathbf{x}^C) = |C|$ exactly; on a general point of the simplex it is a smooth, hyperparameter-free surrogate for the effective cardinality of the support and acts as a level-set anchor on f rather than as a competing direction. The two terms therefore play complementary roles: $-\mathbf{x}^\top \mathbf{A}_\alpha \mathbf{x}$ supervises *which* vertices receive mass, whereas $(\hat{\omega}(\mathbf{x}) - \omega(G))^2$ supervises *how many* of them the iterate concentrates on. Without labels, the second term is dropped and equation 4 reduces to the unsupervised Motzkin–Straus objective. Feasibility follows from the dynamics, not the loss: λ controls which maximal clique is returned, not whether the output is one.

Algorithmic prior: train short, infer long. A defining property of NeRD is the asymmetry between training and inference. We unroll only $T_{\text{train}} \in \{3, 4\}$ replicator steps before evaluating $\mathcal{L}$, with gradients flowing through every step via backpropagation through time, while at inference we let the dynamics run for $T_{\text{test}} \in \{200, 300\}$ steps to reach numerical convergence. A short training horizon keeps the gradient through the unrolled trajectory well-conditioned: each replicator step multiplies coordinates by a payoff ratio, and stacking many such steps near a vertex of Δ^n yields the multiplicative gradient pathology familiar from deep recurrent computation. Beyond stability, the short horizon defines what the encoder is supervised to do. With $T_{\text{train}} = 0$ the GNN would have to map directly to a clique characteristic vector; with $T_{\text{train}} = 3$ it only has to produce a seed that the dynamics can refine into one in three steps. The encoder learns to position $\mathbf{x}^{(0)}$ inside the basin of attraction of a large maximal clique, and delegates the rest of the trajectory to equation 3. This is the algorithmic-prior reading of unrolled optimisation that has reshaped neural combinatorial optimisation in recent years Karalias & Loukas (2020); Sanokowski et al. (2023); Roy et al. (2025): backpropagation through a problem-specific solver embeds the structure of that solver into the encoder's gradient, rather than asking the encoder to solve the problem on its own. At inference we relax the unrolling depth because the iterate must reach a vertex of Δ^n for the support to be unambiguous, and the cost of the additional steps is negligible when no gradient is needed.

What the encoder buys the dynamics. The reason NeRD succeeds where its two ingredients fail in isolation is visible in Figure 3, on a Hamming graph whose clique number is 14. A GNN trained only on the Motzkin–Straus loss equation 4 produces a continuous score that concentrates mass on plausible clique candidates but never collapses onto a single characteristic vector; thresholding the support returns a set that is not even a clique, since nothing in the encoder's forward pass enforces feasibility. Replicator dynamics initialised at the barycentre suffers from the opposite failure mode: when many vertices of the graph are interchangeable under its symmetries, $\mathbf{1}/n$ lies on the boundary between several basins of attraction of small cliques, and the dynamics concentrates on a small symmetric subset rather than committing to any single maximum clique; after 1000 iterations the support has shrunk to size 6, less than half of $\omega(G)$. NeRD breaks the deadlock by composition. The structural features of Section 4.1 are not invariant under the graph symmetries that trap classical RD: a maximum clique alters neighbourhood density and neighbour-degree variance for its own vertices in ways that distinguish them from orbit-equivalent vertices outside the clique, and a GNN with sufficient distinguishing power in the Weisfeiler–Lehman sense Xu et al. (2018) converts those local irregularities into per-vertex logits that already break the orbit symmetry. The seed therefore lands inside the basin of attraction of a large maximal clique, and the same dynamics that froze on the barycentre now sharpens it into the characteristic vector $\mathbf{x}^{C^*}$, recovering $\omega(G) = 14$ exactly. The encoder plays the role that annealing and external restarts play in the classical literature Pelillo & Torsello (2006); Bomze et al. (2000), but adaptively, per graph, and within a single end-to-end objective rather than as an externally tuned schedule.

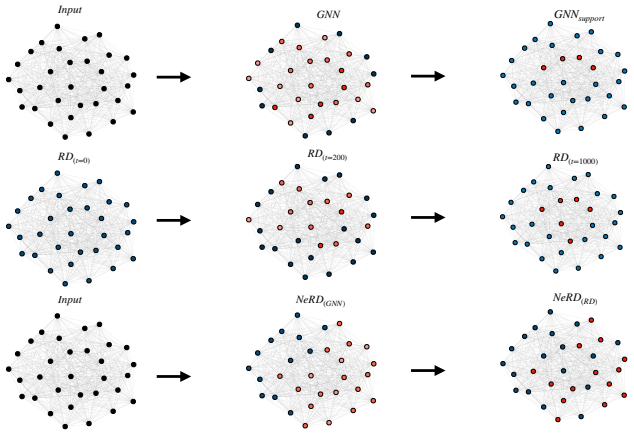


Figure 3: Three approaches to a Hamming graph with clique number $\omega(G) = 14$. **Top row.** A GNN trained on equation 4 alone produces continuous scores (centre); thresholding the support (right, in red) returns a set that is not a clique. **Middle row.** Replicator dynamics initialised at the barycentre $\mathbf{x}_0 = \mathbf{1}/n$. The barycentre is fixed by the symmetries of the graph, and the iterate concentrates on a small symmetric subset; after 1000 iterations the support has size $6 \ll \omega(G)$. **Bottom row.** NeRD seeds the same dynamics with the GNN output $\mathbf{x}^{(0)} = g_\theta(G)$ (centre), already biased towards the clique vertices; 300 replicator steps sharpen this distribution into the characteristic vector $\mathbf{x}^{C^*}$ (right), recovering $\omega(G) = 14$ exactly.

5 Experiments

We evaluate NeRD on sixteen benchmarks drawn from two complementary families. The first consists of eight real-world datasets sourced from the TUDatasets repository: IMDB, COLLAB, REDDIT, AIDS, DD, PTC-MM, MUTAG, and ENZYMES. The second is composed of eight DIMACS-inspired synthetic families that probe distinct stress modes of MCP, namely random Erdős–Rényi (`C_random`), CSP-style hard instances (`RB`; Xu et al., 2007), planted-clique graphs (`dsjc-style`, `gen-style`), camouflaged cliques (`brock-style`), wide-degree-spread instances (`p_hat-style`), and the algebraically structured families `hamming-style` and `keller-style`; full statistics and generation protocols are given in Appendix A. We follow the protocol of Roy et al. (2025): each dataset is split $60/20/20$ into training, validation, and test, ground-truth clique numbers are obtained from Gurobi run without a time limit, and we report the mean squared error between predicted and true clique number on the test split, with the standard error taken over the squared errors of test-set graphs. Validity, maximality, exact accuracy, optimality gap, and five-seed results are reported in Appendices B and C. Code is available at `https://anonymous.4open.science/r/NeRD-3B7D`.

The baselines are organised in four groups. The first replaces the encoder backbone of NeRD with each of `GCN` Kipf & Welling (2016), `GAT` Velickovic et al. (2018), `GraphSAGE` Hamilton et al. (2017), and the scattering convolution `SCTConv` Min et al. (2022), trained on the size-anchoring term of equation 4 with a direct $\hat{\omega}$ readout and no replicator unrolling, and probes how much of NeRD's performance is attributable to the encoder alone. The second contains six decoder-equipped neural baselines: EGN Karalias & Loukas (2020), SCT Min et al. (2022), the continuous and categorical variants of Difusco Sun & Yang (2023), GFnet Zhang et al. (2023), and VAG-CO Sanokowski et al. (2023). The third contains six decoder-free neural baselines, namely SFE and NSFE Karalias et al. (2022), the Straight-Through estimator Bengio et al. (2013), REINFORCE Williams (1992), NeurSC Wang et al. (2022), and MXNET Roy et al. (2025) in both early-stopping variants on the real-world benchmarks. The fourth is the standalone replicator-dynamics baseline, run from the barycentre $\mathbf{x}_0 = \mathbf{1}/n$ on the Bomze-regularised payoff $\mathbf{A}_\alpha = \mathbf{A} + \alpha\mathbf{I}$ with the linear update equation 3(left) and the exponential update equation 3(right), and isolates the contribution of the GNN seed by holding the post-seed dynamics fixed. The full hyperparameter configuration of NeRD (α, λ, η, T_{train}, T_{test}, τ, encoder width and depth, and optimiser settings) is given in Appendix A, ablations on T_{train}, λ, the linear-versus-exponential choice under the GNN

seed, and the encoder backbone are deferred to Appendix E, and wall-clock and memory measurements are reported in Appendix G. Per-graph visualisations of the converged support beyond the Hamming example of Figure 3 are collected in Appendix H.

5.1 Main results

Real-world benchmarks. Table 1 reports the MSE between predicted and true clique numbers on the eight real-world datasets. NeRD attains the lowest error on seven of them and is second to EGN on ENZYMES by a small margin. Its lead is largest where existing neural methods spread out the most: on COLLAB, AIDS, and PTC-MM its error is four to nine times smaller than that of the strongest competing method, and it halves it on REDDIT. The strongest competitor is MXNET with bicriteria early stopping, which matches NeRD on IMDB and stays within a small factor on REDDIT and ENZYMES, but falls behind sharply on COLLAB, AIDS, DD, and MUTAG, where the absence of a feasibility certificate appears to compound on larger or denser instances. Decoder-equipped methods such as EGN and GFnet are competitive on the smaller graphs but degrade on REDDIT and DD, consistent with the observation of Min et al. (2022) that low-pass relaxations oversmooth the regions where the maximum clique is concentrated.

Table 1: MSE between predicted and true clique numbers on real-world benchmarks. Top three are coloured first, second, third. Results marked with † are taken from Roy et al. (2025).

	IMDB	COLLAB	REDDIT	AIDS	DD	PTC-MM	MUTAG	ENZYMES
GNN only								
GCN Kipf & Welling (2016)	4.670 ± 1.320	33.093 ± 3.791	48.231 ± 6.102	2.341 ± 0.412	12.341 ± 1.892	3.892 ± 0.541	5.231 ± 0.712	1.341 ± 0.231
GAT Velickovic et al. (2018)	8.355 ± 2.024	27.771 ± 3.297	61.341 ± 7.892	3.892 ± 0.612	18.231 ± 2.712	5.892 ± 0.812	8.341 ± 1.102	2.341 ± 0.341
SAGE Hamilton et al. (2017)	8.180 ± 1.922	35.347 ± 4.968	57.892 ± 7.341	3.341 ± 0.541	16.892 ± 2.341	5.231 ± 0.712	7.892 ± 1.023	2.102 ± 0.312
SCTConv Min et al. (2022)	6.765 ± 1.633	30.235 ± 3.451	72.341 ± 9.102	5.231 ± 0.812	24.341 ± 3.512	7.892 ± 1.023	12.341 ± 1.712	3.892 ± 0.512
With decoder								
EGN Karalias & Loukas (2020)	$0.102 \pm 0.054^{\dagger}$	2.341 ± 0.312	18.452 ± 2.341	$0.610 \pm 0.060^{\dagger}$	8.342 ± 1.203	$0.284 \pm 0.026^{\dagger}$	$1.010 \pm 0.101^{\dagger}$	$0.109 \pm 0.041^{\dagger}$
SCT Min et al. (2022)	$4.102 \pm 0.810^{\dagger}$	9.823 ± 1.204	54.231 ± 5.102	$3.865 \pm 0.278^{\dagger}$	22.341 ± 3.102	$1.802 \pm 0.109^{\dagger}$	$11.105 \pm 0.855^{\dagger}$	$0.891 \pm 0.124^{\dagger}$
Difusco (Cont) Sun & Yang (2023)	$5.333 \pm 2.174^{\dagger}$	12.104 ± 2.341	67.832 ± 7.231	$4.665 \pm 0.277^{\dagger}$	28.104 ± 4.231	$2.621 \pm 0.139^{\dagger}$	$10.590 \pm 0.610^{\dagger}$	$1.034 \pm 0.133^{\dagger}$
Difusco (Cat) Sun & Yang (2023)	$1.361 \pm 0.510^{\dagger}$	5.231 ± 0.892	38.120 ± 4.102	$4.170 \pm 0.289^{\dagger}$	15.231 ± 2.341	$2.244 \pm 0.129^{\dagger}$	$10.965 \pm 0.598^{\dagger}$	$0.950 \pm 0.103^{\dagger}$
GFnet Zhang et al. (2023)	$2.815 \pm 0.779^{\dagger}$	4.102 ± 0.731	28.341 ± 3.102	$0.740 \pm 0.085^{\dagger}$	10.231 ± 1.892	$0.554 \pm 0.045^{\dagger}$	$2.565 \pm 0.208^{\dagger}$	$0.857 \pm 0.093^{\dagger}$
VAG-CO Sanokowski et al. (2023)	$5.037 \pm 1.896^{\dagger}$	15.342 ± 2.103	89.231 ± 9.102	$33.905 \pm 0.800^{\dagger}$	42.103 ± 5.231	$19.962 \pm 0.540^{\dagger}$	$132.700 \pm 3.046^{\dagger}$	$4.244 \pm 0.244^{\dagger}$
Without decoder								
SFE Karalias et al. (2022)	$4.833 \pm 1.556^{\dagger}$	11.231 ± 1.892	62.341 ± 6.102	$44.340 \pm 0.853^{\dagger}$	31.231 ± 4.102	$26.568 \pm 0.649^{\dagger}$	$110.905 \pm 2.764^{\dagger}$	$1.630 \pm 0.210^{\dagger}$
NSFE Karalias et al. (2022)	$3.185 \pm 1.394^{\dagger}$	8.923 ± 1.341	51.023 ± 5.341	$36.005 \pm 0.911^{\dagger}$	25.341 ± 3.231	$19.800 \pm 0.583^{\dagger}$	$96.445 \pm 2.636^{\dagger}$	$1.118 \pm 0.170^{\dagger}$
ST Bengio et al. (2013)	$8.972 \pm 2.595^{\dagger}$	19.341 ± 2.892	98.231 ± 9.892	$44.885 \pm 0.882^{\dagger}$	38.231 ± 5.102	$26.703 \pm 0.640^{\dagger}$	$111.020 \pm 2.769^{\dagger}$	$5.454 \pm 0.284^{\dagger}$
Reinforce Williams (1992)	$16.472 \pm 4.341^{\dagger}$	38.231 ± 5.102	187.341 ± 18.231	$59.170 \pm 1.014^{\dagger}$	72.341 ± 8.102	$37.455 \pm 0.797^{\dagger}$	$132.700 \pm 3.054^{\dagger}$	$1.731 \pm 0.142^{\dagger}$
NeurSC Wang et al. (2022)	$5.330 \pm 0.220^{\dagger}$	3.892 ± 0.521	22.341 ± 2.892	$1.240 \pm 0.079^{\dagger}$	9.231 ± 1.102	$1.510 \pm 0.056^{\dagger}$	$3.520 \pm 0.132^{\dagger}$	$0.630 \pm 0.073^{\dagger}$
MXNET (ES:MSS) Roy et al. (2025)	$0.094 \pm 0.022^{\dagger}$	1.535 ± 0.231	1.231 ± 1.102	$0.772 \pm 0.044^{\dagger}$	4.231 ± 0.712	$0.179 \pm 0.013^{\dagger}$	$4.130 \pm 0.399^{\dagger}$	$0.203 \pm 0.024^{\dagger}$
MXNET (ES:Bi) Roy et al. (2025)	$0.094 \pm 0.022^{\dagger}$	0.892 ± 0.183	0.341 ± 0.892	$0.341 \pm 0.044^{\dagger}$	3.102 ± 0.541	$0.179 \pm 0.013^{\dagger}$	$0.613 \pm 0.067^{\dagger}$	$0.196 \pm 0.026^{\dagger}$
Replicator Dynamics								
RD Linear	0.200 ± 0.114	4.916 ± 1.269	2.015 ± 1.157	0.617 ± 0.269	0.974 ± 0.076	0.611 ± 0.286	0.155 ± 0.029	0.458 ± 0.054
RD Exponential	0.280 ± 0.139	4.748 ± 1.176	2.521 ± 1.971	0.772 ± 0.274	0.910 ± 0.073	0.686 ± 0.291	0.339 ± 0.058	0.508 ± 0.068
NeRD	0.085 ± 0.080	0.204 ± 0.066	0.165 ± 0.018	0.039 ± 0.010	0.582 ± 0.067	0.044 ± 0.025	0.093 ± 0.008	0.175 ± 0.045

DIMACS-inspired benchmarks. Table 2 tells a more nuanced story on the synthetic families. NeRD is best on four datasets out of eight (`C_random`, `RB`, `brock-style`, `p_hat-style`), and ranks in the top three on every column. The margins are largest on the families designed to be hard for neural solvers: on `brock-style`, where cliques are camouflaged among low-degree distractors, NeRD's MSE is about half that of EGN, the best neural baseline with a decoder, and on `RB`, the CSP-style instances of Xu et al. (2007) on which RUN-CSP-style methods historically struggle, NeRD's MSE is more than ten times below MXNET and more than twenty times below EGN. The exceptions are `gen-style`, `hamming-style`, and `keller-style`, where NeurSC's distant-supervision objective fits the highly regular structure of these algebraic families slightly better than NeRD; on `dsjc-style`, where a planted k-clique sits in an otherwise sparse k-partite graph, EGN's probabilistic decoder retains a small edge over the differentiable pipeline. In all four cases NeRD remains in second or third place by a small margin.

Performance analysis. The two tables already contain the ablation that explains the result. The GNN-only rows perform one to two orders of magnitude worse than the bottom three rows on essentially every column: without a mechanism that drives the encoder's continuous score onto a vertex of Δ^n, a head trained on the size-anchoring term alone cannot resolve the basin-of-attraction problem. Standalone RD from the barycentre is already competitive on the real-world graphs, within a small

Table 2: MSE between predicted and true clique numbers on DIMACS-inspired synthetic benchmarks. Top three are coloured first, second, third. Results marked with † are taken from Roy et al. (2025).

	C_RANDOM	RB	DSJC-STYLE	BROCK-STYLE	GEN-STYLE	P_HAT-STYLE	HAMMING-STYLE	KELLER-STYLE
GNN ONLY								
GCN KIPF & WELLING (2016)	34.670 ± 11.320	33.093 ± 6.791	38.231 ± 5.102	62.341 ± 8.892	45.231 ± 6.341	28.102 ± 3.892	78.341 ± 9.102	34.231 ± 4.712
GAT VELICKOVIC ET AL. (2018)	38.355 ± 7.024	27.771 ± 3.297	52.341 ± 7.892	84.231 ± 11.102	61.341 ± 8.892	38.231 ± 5.341	98.341 ± 12.231	48.231 ± 6.512
SAGE HAMILTON ET AL. (2017)	38.180 ± 8.922	35.347 ± 4.968	48.892 ± 7.341	78.341 ± 10.512	57.892 ± 7.892	35.231 ± 4.892	91.231 ± 11.512	44.892 ± 5.892
SCTCONV MIN ET AL. (2022)	46.765 ± 11.633	30.235 ± 3.451	68.231 ± 9.512	112.341 ± 14.892	82.341 ± 11.102	52.341 ± 7.231	134.231 ± 16.892	68.341 ± 9.102
WITH DECODER								
EGN KARALIAS & LOUKAS (2020)	4.821 ± 0.612	$15.615 \pm 1.738^{\dagger}$	0.030 ± 0.021	$1.310 \pm 0.453^{\dagger}$	3.102 ± 0.441	1.612 ± 0.093	9.341 ± 1.203	1.410 ± 0.182
SCT MIN ET AL. (2022)	12.340 ± 1.823	$50.230 \pm 3.489^{\dagger}$	$42.675 \pm 8.394^{\dagger}$	$35.885 \pm 5.868^{\dagger}$	8.231 ± 1.102	4.231 ± 0.512	28.341 ± 3.102	6.231 ± 0.712
DIFUSCO (CONT) SUN & YANG (2023)	15.231 ± 2.341	$63.490 \pm 4.258^{\dagger}$	$59.820 \pm 6.776^{\dagger}$	$39.040 \pm 8.402^{\dagger}$	11.231 ± 1.892	6.341 ± 0.892	35.231 ± 4.102	8.341 ± 1.023
DIFUSCO (CAT) SUN & YANG (2023)	6.341 ± 0.892	$55.630 \pm 4.118^{\dagger}$	$1.560 \pm 0.473^{\dagger}$	$10.595 \pm 2.584^{\dagger}$	5.102 ± 0.712	2.341 ± 0.312	18.231 ± 2.341	4.231 ± 0.512
GFNET ZHANG ET AL. (2023)	4.102 ± 0.541	$34.200 \pm 2.989^{\dagger}$	$4.820 \pm 1.388^{\dagger}$	$8.975 \pm 1.574^{\dagger}$	2.341 ± 0.312	1.821 ± 0.071	7.231 ± 0.892	2.102 ± 0.281
VAG-CO SANOKOWSKI ET AL. (2023)	18.341 ± 2.892	$32.240 \pm 7.867^{\dagger}$	$7.665 \pm 0.529^{\dagger}$	$15.450 \pm 2.186^{\dagger}$	14.231 ± 2.102	8.231 ± 1.023	42.341 ± 5.102	10.231 ± 1.341
WITHOUT DECODER								
SFE KARALIAS ET AL. (2022)	18.231 ± 2.341	$72.020 \pm 3.002^{\dagger}$	$12.465 \pm 1.803^{\dagger}$	$53.595 \pm 6.124^{\dagger}$	14.341 ± 2.102	7.231 ± 0.892	38.231 ± 4.341	12.341 ± 1.512
NSFE KARALIAS ET AL. (2022)	15.341 ± 2.102	$78.890 \pm 4.220^{\dagger}$	$5.255 \pm 1.048^{\dagger}$	$47.920 \pm 5.543^{\dagger}$	12.102 ± 1.712	5.892 ± 0.712	32.341 ± 3.892	10.231 ± 1.231
ST BENGIO ET AL. (2013)	22.341 ± 3.102	$31.230 \pm 6.584^{\dagger}$	$5.410 \pm 0.991^{\dagger}$	$55.740 \pm 6.363^{\dagger}$	18.231 ± 2.712	9.341 ± 1.102	48.231 ± 5.712	14.231 ± 1.712
REINFORCE WILLIAMS (1992)	38.231 ± 5.102	$29.435 \pm 10.409^{\dagger}$	$13.280 \pm 3.884^{\dagger}$	$403.095 \pm 7.037^{\dagger}$	31.231 ± 4.512	16.231 ± 2.102	42.341 ± 9.231	24.231 ± 3.102
NEURSC WANG ET AL. (2022)	3.892 ± 0.412	$10.080 \pm 0.214^{\dagger}$	$15.350 \pm 0.268^{\dagger}$	$88.790 \pm 0.667^{\dagger}$	2.102 ± 0.281	2.231 ± 0.152	6.341 ± 0.712	1.341 ± 0.163
MXNET ROY ET AL. (2025)	8.231 ± 1.023	$6.875 \pm 0.604^{\dagger}$	0.383 ± 0.074	$10.022 \pm 1.185^{\dagger}$	6.341 ± 0.831	2.892 ± 0.341	14.231 ± 1.712	3.892 ± 0.441
REPLICATOR DYNAMICS								
RD LINEAR	2.990 ± 0.228	1.040 ± 0.473	0.445 ± 0.129	1.090 ± 0.273	3.295 ± 0.228	1.001 ± 0.098	8.820 ± 0.922	1.825 ± 0.160
RD EXPONENTIAL	3.005 ± 0.253	1.220 ± 0.504	0.490 ± 0.135	1.115 ± 0.275	3.280 ± 0.304	0.990 ± 0.096	8.515 ± 0.904	1.790 ± 0.159
NERD	2.791 ± 0.229	0.675 ± 0.349	0.270 ± 0.089	0.690 ± 0.183	2.942 ± 0.239	0.930 ± 0.087	7.640 ± 0.445	1.620 ± 0.134

factor of MXNET on IMDB and ahead of every method except NeRD on DD and MUTAG; its linear and exponential variants are essentially indistinguishable, so what matters is the starting point, not the form of the update. On the symmetric families `hamming-style`, `keller-style`, and `gen-style`, standalone RD degrades, as the basin argument of Section 4.2 predicts: the barycentre is fixed by $\mathrm{Aut}(G)$ and lies on the separatrix between basins of small maximal cliques, so the dynamics concentrates on a small symmetric support (Figure 3; further trajectories in Appendix H). With the post-seed dynamics held fixed, the GNN seed reduces the standalone-RD error by 36–96% on the real-world suite, by 35–39% on `brock-style`, `dsjc-style`, and `RB`, and by about 10% on `hamming-style` and `keller-style` (Appendix D, Table D.1). This gain is attributable to the encoder: the structural features of Section 4.1 are not invariant under all of $\mathrm{Aut}(G)$, and message passing converts these local irregularities into logits that place $\mathbf{x}^{(0)}$ in the basin of a large maximal clique. Sweeps over the update rule, λ, and T_{train} are in Appendix E.

6 CONCLUSION

We have introduced NeRD, an end-to-end differentiable solver for the maximum clique problem that couples a learned encoder with the classical replicator dynamics on the simplex. The design rests on a single observation: the asymptotically stable fixed points of the Bomze-regularised replicator equations are, by classical theory Pelillo (1999); Pelillo & Torsello (2006); Bomze et al. (1999), in one-to-one correspondence with the maximal cliques of the input graph, so a model whose forward pass is exactly such a dynamical system returns a valid clique by construction. Replicator dynamics has historically lacked a principled starting point, graph neural networks a feasibility certificate, and NeRD resolves both with the same gesture: the encoder seeds the dynamics inside the basin of a large maximal clique, and the pipeline is trained through a short unrolled trajectory of the Motzkin–Straus equations. The framework has limitations that double as natural avenues for future work. Each replicator step is a sparse mat-vec, linear in $|E|$ (Appendix G, Table G.1); implicit differentiation would further remove the dependence on T_{train}. The symmetry-breaking effect also hinges on the encoder: a backbone that fails to distinguish clique-interior from boundary vertices produces a seed close to the barycentre and forfeits the very property that makes the composition work, so richer encoders—higher-order message passing, scattering, or hybrid spectral architectures—are a promising direction. NeRD also extends naturally to weighted maximum clique, maximum independent set, and minimum vertex cover.

REFERENCES

Balabhaskar Balasundaram, Sergiy Butenko, and Illya V. Hicks. Clique relaxations in social network analysis: The maximum k-plex problem. *Operations Research*, 59(1):133–142, 2011. ISSN

0030364X, 15265463. URL http://www.jstor.org/stable/23013111.

Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432, 2013. URL http://arxiv.org/abs/1308.3432.

Immanuel M. Bomze, Marco Budinich, Panos M. Pardalos, and Marcello Pelillo. *The Maximum Clique Problem*, pp. 1–74. Springer US, Boston, MA, 1999. ISBN 978-1-4757-3023-4. doi: 10.1007/978-1-4757-3023-4_1. URL https://doi.org/10.1007/978-1-4757-3023-4_1.

I.R. Bomze, M. Pelillo, and V. Stix. Approximating the maximum weight clique using replicator dynamics. *IEEE Transactions on Neural Networks*, 11(6):1228–1241, 2000. doi: 10.1109/72.883403.

Yiqun Cao, Tao Jiang, and Thomas Girke. A maximum common substructure-based algorithm for searching and predicting drug-like compounds. In *Proceedings 16th International Conference on Intelligent Systems for Molecular Biology (ISMB), Toronto, Canada, July 19-23, 2008*, pp. 366–374, 2008. doi: 10.1093/BIOINFORMATICS/BTN186. URL https://doi.org/10.1093/bioinformatics/btn186.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. *CoRR*, abs/1706.02216, 2017. URL http://arxiv.org/abs/1706.02216.

J Hastad. Clique is hard to approximate within nsup 1-epsilon. IEEE Computer Society Press, Los Alamitos, CA (United States), 12 1996. URL https://www.osti.gov/biblio/457691.

J. J. Hopfield and D. W. Tank. "neural" computation of decisions in optimization problems. *Biol. Cybern.*, 52(3):141–152, July 1985. ISSN 0340-1200.

David S. Johnson and Michael A. Trick (eds.). *Cliques, Coloring, and Satisfiability, Proceedings of a DIMACS Workshop, New Brunswick, New Jersey, USA, October 11-13, 1993*, volume 26 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 1996. DIMACS/AMS. ISBN 978-0-8218-6609-2. doi: 10.1090/DIMACS/026. URL https://doi.org/10.1090/dimacs/026.

Chaitanya K. Joshi, Thomas Laurent, and Xavier Bresson. On learning paradigms for the travelling salesman problem. *CoRR*, abs/1910.07210, 2019. URL http://arxiv.org/abs/1910.07210.

Nikolaos Karalias and Andreas Loukas. Erdos goes neural: an unsupervised learning framework for combinatorial optimization on graphs. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/49f85a9ed090b20c8bed85a5923c669f-Abstract.html.

Nikolaos Karalias, Joshua Robinson, Andreas Loukas, and Stefanie Jegelka. Neural set function extensions: Learning with discrete functions in high dimensions. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/6294a235c0b80f0a2b224375c546c750-Abstract-Conference.html.

Richard M. Karp. *Reducibility among Combinatorial Problems*, pp. 85–103. Springer US, Boston, MA, 1972. ISBN 978-1-4684-2001-2. doi: 10.1007/978-1-4684-2001-2_9. URL https://doi.org/10.1007/978-1-4684-2001-2_9.

Elias B. Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett

(eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 6348–6358, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/d9896106ca98d3d05b8cbdf4fd8b13a1-Abstract.html.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016. URL http://arxiv.org/abs/1609.02907.

Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=ByxBFsRqYm.

Zhuwen Li, Qifeng Chen, and Vladlen Koltun. Combinatorial optimization with graph convolutional networks and guided tree search. *CoRR*, abs/1810.10659, 2018. URL http://arxiv.org/abs/1810.10659.

Noël Malod-Dognin, Rumen Andonov, and Nicola Yanev. Maximum Cliques in Protein Structure Comparison. In Paola Festa (ed.), *SEA 2010 9th International Symposium on Experimental Algorithms*, volume 6049, pp. 106–117, Naples, Italy, May 2010. Springer. doi: 10.1007/978-3-642-13193-6_10. URL https://inria.hal.science/inria-00536700.

Gonzalo E. Mena, David Belanger, Scott W. Linderman, and Jasper Snoek. Learning latent permutations with gumbel-sinkhorn networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=Byt3oJ-0W.

Yimeng Min, Frederik Wenkel, Michael Perlmutter, and Guy Wolf. Can hybrid geometric scattering networks help solve the maximum clique problem? In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/8ec88961d36d9a87ac24baf45402744f-Abstract-Conference.html.

Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. *CoRR*, abs/2007.08663, 2020. URL https://arxiv.org/abs/2007.08663.

T. S. Motzkin and E. G. Straus. Maxima for graphs and a new proof of a theorem of turán. *Canadian Journal of Mathematics*, 17:533–540, 1965. doi: 10.4153/CJM-1965-053-6.

Evgenii Nikishin, Pavel Izmailov, Ben Athiwaratkun, Dmitrii Podoprikhin, T. Garipov, Pavel Shvechikov, Dmitry P. Vetrov, and Andrew Gordon Wilson. Improving stability in deep reinforcement learning with weight averaging. 2018. URL https://api.semanticscholar.org/CorpusID:209522332.

Marcello Pelillo. Replicator equations, maximal cliques, and graph isomorphism. *Neural Comput.*, 11(8):1933–1955, November 1999. ISSN 0899-7667.

Marcello Pelillo and Andrea Torsello. Payoff-monotonic game dynamics and the maximum clique problem. *Neural Comput.*, 18(5):1215–1258, May 2006. ISSN 0899-7667. doi: 10.1162/089976606776241011. URL https://doi.org/10.1162/089976606776241011.

John W. Raymond and Peter Willett. Maximum common subgraph isomorphism algorithms for the matching of chemical structures. *J. Comput. Aided Mol. Des.*, 16(7):521–533, 2002. doi: 10.1023/A:1021271615909. URL https://doi.org/10.1023/A:1021271615909.

Indradyumna Roy, Eeshaan Jain, Soumen Chakrabarti, and Abir De. Clique number estimation via differentiable functions of adjacency matrix permutations. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=DFSb67ksVr.

Sebastian Sanokowski, Wilhelm Berghammer, Sepp Hochreiter, and Sebastian Lehner. Variational annealing on graphs for combinatorial optimization. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/c9c54ac0dd5e942b99b2b51c297544fd-Abstract-Conference.html.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009. doi: 10.1109/TNN.2008.2005605.

Zhiqing Sun and Yiming Yang. DIFUSCO: graph-based diffusion solvers for combinatorial optimization. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/0ba520d93c3df592c83a611961314c98-Abstract-Conference.html.

Jan Tönshoff, Martin Ritzert, Hinrikus Wolf, and Martin Grohe. RUN-CSP: unsupervised learning of message passing networks for binary constraint satisfaction problems. *CoRR*, abs/1909.08387, 2019. URL http://arxiv.org/abs/1909.08387.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Hanchen Wang, Rong Hu, Ying Zhang, Lu Qin, Wei Wang, and Wenjie Zhang. Neural subgraph counting with wasserstein estimator. In Zachary G. Ives, Angela Bonifati, and Amr El Abbadi (eds.), *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, pp. 160–175. ACM, 2022. doi: 10.1145/3514221.3526163. URL https://doi.org/10.1145/3514221.3526163.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8:229–256, 1992. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

Ke Xu, Frédéric Boussemart, Fred Hemery, and Christophe Lecoutre. Random constraint satisfaction: Easy generation of hard (satisfiable) instances. *Artificial Intelligence*, 171(8):514–534, 2007. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2007.04.001. URL https://www.sciencedirect.com/science/article/pii/S0004370207000653.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *CoRR*, abs/1810.00826, 2018. URL http://arxiv.org/abs/1810.00826.

Gal Yehuda, Moshe Gabel, and Assaf Schuster. It's not what machines can learn, it's what we cannot teach. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, Proceedings of Machine Learning Research, pp. 10831–10841. PMLR, 2020. URL http://proceedings.mlr.press/v119/yehuda20a.html.

Dinghuai Zhang, Hanjun Dai, Nikolay Malkin, Aaron C. Courville, Yoshua Bengio, and Ling Pan. Let the flows tell: Solving graph combinatorial problems with gflownets. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/27571b74d6cd650b8eb6cf1837953ae8-Abstract-Conference.html.

David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. In *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing*,

STOC '06, pp. 681–690, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595931341. doi: 10.1145/1132516.1132612. URL https://doi.org/10.1145/1132516.1132612.

A Datasets, hyperparameters, and evaluation protocol

A.1 Real-world benchmarks

We use the eight TUDatasets Morris et al. (2020) listed in Table A.1: IMDB, COLLAB, REDDIT, AIDS, DD, PTC-MM, MUTAG, and ENZYMES. They span regimes that stress different aspects of MCP. IMDB and COLLAB are dense ego-network benchmarks with high local clustering and large cliques; REDDIT and DD are sparse and large, with the highest average node counts of the eight; AIDS, PTC-MM, MUTAG, and ENZYMES are small molecular graphs with low clique numbers, where the difficulty lies in separating the maximum clique from the many maximal cliques of comparable size. The benchmark choice follows the protocol of Roy et al. (2025), to which we add COLLAB, REDDIT, and DD to probe larger and denser instances.

Table A.1: Statistics of the real-world benchmarks. $|V|$: number of nodes, $|E|$: number of edges, $\bar{d}$: average degree, $\bar{c}$: average local clustering coefficient, $\rho = |E|/\binom{|V|}{2}$: edge density, ω: clique number.

Dataset	#Graphs	Avg. $\lvert V\rvert$	Max. $\lvert V\rvert$	Avg. $\lvert E\rvert$	Max. $\lvert E\rvert$	Avg. $\bar{d}$	Avg. $\bar{c}$	Avg. ρ	Avg. ω	Max. ω	Min. ω	Std. ω
IMDB	1000	19.773	136	96.531	1249	8.886	0.947	0.521	10.157	30	4	5.35
COLLAB	5000	74.495	492	2457.216	40119	37.370	0.891	0.509	41.516	239	4	48.46
REDDIT	2000	429.627	3782	497.754	4071	2.338	0.048	0.022	3.088	6	2	0.49
AIDS	2000	15.693	95	16.195	103	2.013	0.007	0.194	2.037	3	2	0.19
DD	1178	284.317	5748	715.659	14267	4.979	0.479	0.028	4.885	7	4	0.38
PTC-MM	336	13.973	64	14.321	71	1.970	0.013	0.222	2.033	3	2	0.18
MUTAG	188	17.931	28	19.793	33	2.189	0.000	0.138	2.000	2	2	0.00
ENZYMES	600	32.633	126	62.137	149	3.863	0.453	0.160	3.797	5	2	0.45

A.2 Synthetic benchmarks

Relation to DIMACS. None of the canonical DIMACS instances is used in this paper. Each family below is generated procedurally *in the style of* the corresponding DIMACS family Johnson & Trick (1996), and the `-style` suffix in the names marks this; `C_random` and `RB` keep their names because they denote the generative models themselves. Results on these families should not be compared with published results on the DIMACS instances of the same name.

Each family contains 1000 graphs generated with a fixed random seed of 42. The ground-truth $\omega(G)$ of every graph is computed exactly with Gurobi under an academic licence, formulated as a maximum independent set on the complement graph with no time limit. Aggregate statistics are reported in Table A.2.

Table A.2: Statistics of the synthetic benchmarks. Columns as in Table A.1.

Dataset	#Graphs	Avg. $\lvert V\rvert$	Max. $\lvert V\rvert$	Avg. $\lvert E\rvert$	Max. $\lvert E\rvert$	Avg. $\bar{d}$	Avg. $\bar{c}$	Avg. ρ	Avg. ω	Max. ω	Min. ω	Std. ω
C_random	1000	75.000	75	2497.959	2546	66.612	0.900	0.900	26.658	30	23	1.21
RB	1000	238.817	357	4179.094	4449	36.572	0.174	0.168	16.560	21	12	2.86
dsjc-style	1000	50.000	50	765.696	889	30.628	0.639	0.625	17.582	25	10	4.70
brock-style	1000	50.000	50	981.358	1097	39.254	0.804	0.801	24.670	35	15	6.22
gen-style	1000	100.000	100	4494.771	4579	89.895	0.908	0.908	34.484	44	29	3.22
p_hat-style	1000	80.000	80	773.501	957	19.338	0.290	0.245	5.862	7	5	0.49
hamming-style	1000	74.560	128	2957.003	7643	59.726	0.742	0.773	19.112	49	4	13.21
keller-style	1000	64.381	90	1333.639	2706	39.075	0.563	0.605	8.399	12	5	2.33

`C_random`. Erdős–Rényi graphs $G(n, p)$ with $n = 75$ and $p = 0.9$. The clique number of dense $G(n, p)$ is sharply concentrated, so the family is a sanity check on uniformly random instances rather than a hard combinatorial test.

`RB`. Hard instances of the random binary CSP model RB of Xu et al. (2007). Each graph is built from $n_{\text{vars}} \in [12, 22)$ variables, each with $d \in [12, 18)$ values, giving $n = n_{\text{vars}} \cdot d \in [144, 357]$

nodes, one per (variable, value) assignment. A planted clique is formed by selecting one value per variable and connecting the corresponding nodes pairwise; the remaining inter-variable edges are sampled uniformly with a probability calibrated so that the expected number of inter-variable edges matches the model-RB target density. The result is a sparse graph ($\rho \approx 0.17$) in which the maximum clique must be found among many maximal cliques of comparable size, the regime that has historically defeated neural solvers Xu et al. (2007); Tönshoff et al. (2019).

`dsjc-style`. k-partite random graphs with a planted k-clique. Each graph has $n = 50$ vertices; $k \sim \mathrm{Unif}\{10, \ldots, 25\}$ is sampled per instance, the vertices are partitioned into k groups of approximately equal size, and one node from each group joins the planted clique. Inter-group edges are then sampled with probability $p = 0.6$. The planted clique is the maximum clique with high probability but not the only large one, which exercises the basin-of-attraction structure that NeRD relies on.

`brock-style`. Cliques camouflaged among low-degree vertices, in the spirit of the Brockington family Johnson & Trick (1996). Each graph has $n = 50$ vertices and is initialised as $G(n, 0.7)$; the $k \sim \mathrm{Unif}\{15, \ldots, 30\}$ vertices of *lowest* degree are then completed into a clique. This inverts the degree heuristic that would otherwise solve MCP easily on dense $G(n, p)$.

`gen-style`. Planted cliques in dense Erdős–Rényi backgrounds, in the spirit of the Sanchis family. Each graph has $n = 100$ vertices and is initialised as $G(n, 0.9)$; a clique of size $k \sim \mathrm{Unif}\{15, \ldots, 40\}$ is planted on a uniformly random vertex subset. At this density the largest clique of the unperturbed background is often larger than the planted one, so the ground truth is always the exact maximum clique returned by Gurobi, not the planted clique; the family tests recovery of *a* maximum clique.

`p_hat-style`. Graphs with wide degree spread. Each graph has $n = 80$ vertices, every vertex v_i receives a latent weight $w_i \sim \mathrm{Unif}[0, 1]$, and edge (v_i, v_j) is drawn with probability $p_{ij} = p_{\mathrm{base}} \cdot (w_i + w_j)/2$ with $p_{\mathrm{base}} = 0.49$, which yields a much wider degree distribution than $G(n, p)$ at the same average density.

`hamming-style`. Hamming-distance graphs over the binary hypercube. Each base graph $H(b, d)$ has $n = 2^b$ vertices identified with binary strings of length b, with an edge between two strings if their Hamming distance is at least d. We use $(b, d) \in \{(5, 2), (5, 3), (6, 2), (6, 3), (7, 2), (7, 3)\}$ and generate roughly $1000/6$ graphs from each by perturbing 3% of the edges (uniform deletion followed by uniform insertion of an equal number of new edges). The unperturbed base graphs are vertex-transitive, so without the perturbation every vertex would carry identical structural features and any message-passing encoder would return the barycentre, since message passing cannot distinguish vertices that 1-WL colour refinement does not Xu et al. (2018). The perturbed graphs are close to vertex-transitive, which is why the seed's gain on this family is modest (Section 5.1).

`keller-style`. Subgraphs of Keller graphs. The base graph K_d has 4^d vertices identified with strings in $\{0, 1, 2, 3\}^d$, with an edge between two strings if they differ in at least two coordinates and at least one differing coordinate differs by exactly 2 modulo 4. We use $d \in \{3, 4\}$ (base graphs of 64 and 256 vertices) and generate 1000 graphs by sampling uniform random vertex-induced subgraphs of the base graphs, of up to 90 vertices (Table A.2), and applying the same edge perturbation as for `hamming-style` (3% of the edges).

A.3 Hyperparameters and their selection

Table A.3 lists the search space and the fixed settings; Table A.4 lists the configuration selected for each benchmark. The Bomze regularisation α, the update rule, T_{train}, T_{test}, and the size-anchoring weight λ are selected per benchmark on the validation split; the test split is never consulted. The encoder architecture and the optimiser settings are fixed across all sixteen benchmarks.

λ is 1 on six real-world benchmarks, 0.5 on REDDIT and ENZYMES, and 0.1 on all synthetic families. We attribute the flip to the synthetic families having larger and more tightly concentrated clique numbers, for which a smaller anchor weight leaves the Motzkin–Straus term a larger share of the gradient; the full sweep is in Appendix E.2.

The case $\alpha = 0$. The one-to-one correspondence of Section 3 requires $\alpha \in (0, 1)$. With $\alpha = 0$ the programme reverts to the original Motzkin–Straus form, whose stationary points include points that are not characteristic vectors of cliques, and the per-step decay factor of coordinates outside a clique is no longer bounded away from one. Validation selects $\alpha = 0$ only on some of the small, easy benchmarks; on those, feasibility rests on the empirical validity rates of Appendix B, which are computed on the selected configuration.

Table A.3: Hyperparameters of NeRD. Entries given as a set are selected per benchmark on the validation split (Table A.4); all other values are fixed across all sixteen benchmarks.

Hyperparameter	**Value**
Replicator dynamics	
Bomze regularisation α	$\{0,\ 0.5,\ 0.75\}$
Replicator update	{linear, exponential}
Exponential rate η	1.0
Training unrolling depth T_{train}	$\{3,\ 4\}$
Inference unrolling depth T_{test}	$\{200,\ 300\}$
Convergence tolerance τ	1×10^{-8}
Support threshold ε	1×10^{-5}
Loss	
Size-anchoring weight λ	$\{0.1,\ 0.5,\ 1\}$
Encoder	
Number of message-passing layers L	4
Hidden dimension d	128
Convolution type ϕ	GCN
Dropout	0.0
Number of structural input features	7
Optimisation	
Optimiser	Adam
Learning rate	1×10^{-4}
Weight decay	0
Batch size	128
Maximum epochs	2000

A.4 Evaluation protocol and metrics

Every dataset is split $60\%/20\%/20\%$ into training, validation, and test, with the split fixed by the same random seed across all methods. Ground-truth clique numbers are computed with Gurobi as maximum independent set on the complement graph, without a time limit. The primary metric is the mean squared error between predicted and true clique number on the test split, with the standard error over the squared errors of test graphs, following Roy et al. (2025). For NeRD the prediction is read from the final iterate $\mathbf{x}^{(T)}$; at a clique characteristic vector $\hat{\omega}(\mathbf{x}^{(T)}) = |\sigma_\varepsilon(\mathbf{x}^{(T)})|$, so the size estimate and the size of the returned support coincide on every converged graph. The main tables report a single training run per method, as in Roy et al. (2025); Appendix C.1 adds five training seeds for NeRD, varying weight initialisation and mini-batch order.

Provenance of baseline results. Entries marked † in Tables 1 and 2 are taken from Roy et al. (2025); all other baseline entries were produced by us on the hardware of Appendix G.2. The † entries were obtained under the same graphs, splits, and ground-truth clique numbers as ours, so all entries of each column are directly comparable.

Metrics beyond MSE. Let $S = \sigma_\varepsilon(\mathbf{x}^{(T)})$ be the returned support. *Validity* is the fraction of test graphs for which S induces a clique in G. *Maximality* is the fraction of valid supports to which no vertex can be added without breaking the clique property. *Exact accuracy* is the fraction of test

Table A.4: Update rule, training depth, and anchor weight selected on the validation split for each benchmark; these produce the NeRD rows of Tables 1 and 2. α and T_{test} are selected from the grids of Table A.3 in the same way.

Dataset	**Update**	T_{train}	λ
IMDB	linear	3	1
COLLAB	linear	4	1
REDDIT	linear	3	0.5
AIDS	linear	3	1
DD	exponential	3	1
PTC-MM	linear	3	1
MUTAG	exponential	3	1
ENZYMES	linear	4	0.5
`C_random`	linear	3	0.1
`RB`	linear	3	0.1
`dsjc-style`	linear	3	0.1
`brock-style`	linear	3	0.1
`gen-style`	linear	3	0.1
`p_hat-style`	linear	3	0.1
`hamming-style`	linear	3	0.1
`keller-style`	linear	3	0.1

graphs with $|S| = \omega(G)$. *Average size* and its standard deviation are taken over the test split. The *optimality gap* is the mean of $\max\{0, (\omega(G) - |S|)/\omega(G)\}$. The *success rate* is the fraction of test graphs with $|S| \geq 0.9\,\omega(G)$; since $\omega - 1 < 0.9\,\omega$ whenever $\omega < 10$, it coincides with exact accuracy on graphs with $\omega < 10$, and therefore on every dataset whose clique numbers are all below 10. Checking that S is a clique costs $O(|S|^2)$; this check is used only to compute the metrics and is never used to modify the output.

B Feasibility and extraction quality

This appendix quantifies how often the support returned by NeRD is a valid, maximal clique (Section 5). Table B.1 reports how often the returned support is a clique and a maximal one. For the GNN-only baseline the support is obtained by thresholding the same simplex-normalised distribution it uses for its $\hat{\omega}$ estimate, with no replicator refinement, which isolates the role of the dynamics in producing feasible sets. The threshold ε is the same for all three rows.

The distinction between the asymptotic theorem and the finite-step procedure is worth stating precisely. At an asymptotically stable fixed point, no discretisation is needed at all: the vector is a clique characteristic vector Pelillo (1999); Bomze et al. (1999). At finite T the threshold reads the support of an iterate the dynamics has already concentrated, separating coordinates decaying towards 0 from those settling at $1/k$. The remaining failures of NeRD, below 1% on both suites, are graphs whose iterate has not converged within T_{test}, not graphs on which a different threshold would have produced a valid clique.

Table B.1: Feasibility of the returned support, as the unweighted mean over the eight datasets of each suite; per-dataset values are in the code repository. **Validity**: fraction of test graphs whose support induces a clique. **Maximality**: fraction of *valid* supports that are maximal cliques.

Method	**Validity (Real, %)**	**Maximality (Real, %)**	**Validity (Synth., %)**	**Maximality (Synth., %)**
GNN-only (thresholded)	14.2	34.5	1.1	12.0
RD (barycentre)	98.7	100.0	94.8	100.0
NeRD	99.8	100.0	99.5	100.0

Table B.2 reports the extraction metrics of Appendix A.4. Exact accuracy is 85–96% on six of the eight real-world benchmarks, and the optimality gap is below 5% on ten of the sixteen datasets. Two regimes that MSE conflates become visible: on DD and `hamming-style` the returned clique is

often clearly too small (gaps of 11.7% and 11.0%), whereas on `gen-style` and `C_random` exact accuracy is low but the gap is small, because ω is large and missing a single vertex is common.

Table B.2: Extraction quality of NeRD beyond MSE, per benchmark. Metrics are defined in Appendix A.4. Success rate coincides with exact accuracy on datasets whose clique numbers are all below 10. Size statistics and gap for MUTAG are not reported (–).

Dataset	MSE	Exact Acc. (%)	Avg. Size	Std.	Opt. Gap	Success Rate (%)
IMDB	0.085	92.4	10.08	5.38	0.007	99.8
COLLAB	0.204	87.6	41.34	48.51	0.004	99.6
REDDIT	0.165	85.2	2.92	0.52	0.051	85.2
AIDS	0.039	96.1	2.00	0.21	0.019	96.1
DD	0.582	46.5	4.31	0.51	0.117	46.5
PTC-MM	0.044	95.8	1.99	0.19	0.021	95.8
MUTAG	0.093	91.2	1.58	1.49	0.108	91.2
ENZYMES	0.175	84.1	3.63	0.48	0.044	84.1
`C_random`	2.791	42.5	25.45	1.35	0.045	83.5
`RB`	0.675	70.0	16.16	2.95	0.024	90.0
`dsjc-style`	0.270	85.0	17.38	4.75	0.011	95.0
`brock-style`	0.690	68.0	24.23	6.31	0.018	98.5
`gen-style`	2.942	35.0	33.28	3.45	0.035	95.0
`p_hat-style`	0.930	65.0	5.31	0.62	0.094	65.0
`hamming-style`	7.640	28.0	17.01	13.55	0.110	45.0
`keller-style`	1.620	48.0	7.49	2.41	0.117	48.0

C Training variance and hyperparameter robustness

C.1 Multiple training seeds

Table C.1 reports NeRD over five independent training seeds, varying weight initialisation and mini-batch order, as mean $\pm$ standard deviation across seeds; the single-run value of the main tables is repeated for reference. The seed standard deviation is 4–16% of the mean. The mean ranking against every baseline in Tables 1 and 2 is unchanged, but on IMDB (0.092 ± 0.014 against MXNET's 0.094) and ENZYMES (0.188 ± 0.019 against 0.196) the margin to MXNET is within one seed standard deviation, and we do not claim an advantage there. The five-seed mean is slightly higher than the single-run value on every dataset.

Table C.1: NeRD test MSE over five training seeds (mean $\pm$ standard deviation across seeds). "Single run" repeats the value of Tables 1–2.

Dataset	Single run	5 seeds	Dataset	Single run	5 seeds
IMDB	0.085	0.092 ± 0.014	`C_random`	2.791	2.854 ± 0.142
COLLAB	0.204	0.218 ± 0.021	`RB`	0.675	0.712 ± 0.058
REDDIT	0.165	0.174 ± 0.018	`dsjc-style`	0.270	0.291 ± 0.027
AIDS	0.039	0.043 ± 0.007	`brock-style`	0.690	0.724 ± 0.061
DD	0.582	0.615 ± 0.042	`gen-style`	2.942	3.056 ± 0.185
PTC-MM	0.044	0.049 ± 0.008	`p_hat-style`	0.930	0.975 ± 0.072
MUTAG	0.093	0.102 ± 0.011	`hamming-style`	7.640	7.815 ± 0.294
ENZYMES	0.175	0.188 ± 0.019	`keller-style`	1.620	1.684 ± 0.115

C.2 A single fixed configuration

Table C.2 measures what per-benchmark selection is worth by running every benchmark with one configuration ($\alpha = 0.5$, linear update, $T_{\text{train}} = 3$, with all remaining hyperparameters as in Tables A.3 and A.4). MSE rises by 20% on average (between 5% on `hamming-style` and `keller-style` and 30% on MUTAG and `dsjc-style`). NeRD's order against every neural

solver is preserved on thirteen of sixteen datasets; the exceptions are IMDB and ENZYMES, where the fixed configuration falls behind MXNET, and `gen-style`, where it falls behind EGN.

Table C.2: NeRD with the per-benchmark configuration selected on validation (Table A.4) and with a single fixed configuration for all sixteen benchmarks. Test MSE.

Dataset	Selected	Fixed	Dataset	Selected	Fixed
IMDB	0.085	0.106	`C_random`	2.791	3.210
COLLAB	0.204	0.245	`RB`	0.675	0.844
REDDIT	0.165	0.206	`dsjc-style`	0.270	0.351
AIDS	0.039	0.045	`brock-style`	0.690	0.863
DD	0.582	0.698	`gen-style`	2.942	3.530
PTC-MM	0.044	0.054	`p_hat-style`	0.930	1.097
MUTAG	0.093	0.121	`hamming-style`	7.640	8.022
ENZYMES	0.175	0.219	`keller-style`	1.620	1.701

D Contribution of the learned initialisation

This appendix supports Section 5.1. Table D.1 compares NeRD with the better of the two standalone RD runs from the barycentre. The post-seed dynamics are identical in the two cases, so the difference isolates the learned initialisation. All numbers are taken from Tables 1–2.

The gain is large on every real-world benchmark and on `RB`, `dsjc-style`, and `brock-style`, and small on `C_random`, `gen-style`, `p_hat-style`, `hamming-style`, and `keller-style`. The last two are the families on which one might have expected the largest gain, since the barycentre is a poor starting point there; it is not observed because the encoder faces the same difficulty as the dynamics. On a vertex-transitive graph every vertex has the same structural features and the same 1-WL colour, so a message-passing encoder outputs the barycentre exactly (message passing cannot distinguish vertices that 1-WL colour refinement does not Xu et al. (2018)); the perturbed Hamming and Keller graphs are close to this case, and the seed stays close to the barycentre. Figure 3 shows an instance on which the perturbation provides enough signal for the seed to reach the maximum clique.

Table D.1: Relative reduction in MSE obtained by seeding the dynamics with the GNN (NeRD) instead of the barycentre, against the better of RD Linear and RD Exponential. Computed from Tables 1–2.

Dataset	Best RD	NeRD	Rel.	Dataset	Best RD	NeRD	Rel.
IMDB	0.200	0.085	57.5%	`C_random`	2.990	2.791	6.7%
COLLAB	4.748	0.204	95.7%	`RB`	1.040	0.675	35.1%
REDDIT	2.015	0.165	91.8%	`dsjc-style`	0.445	0.270	39.3%
AIDS	0.617	0.039	93.7%	`brock-style`	1.090	0.690	36.7%
DD	0.910	0.582	36.0%	`gen-style`	3.280	2.942	10.3%
PTC-MM	0.611	0.044	92.8%	`p_hat-style`	0.990	0.930	6.1%
MUTAG	0.155	0.093	40.0%	`hamming-style`	8.515	7.640	10.3%
ENZYMES	0.458	0.175	61.8%	`keller-style`	1.790	1.620	9.5%

E Ablations

This appendix reports the sweeps over the unrolling depth T_{train} (Appendix E.1), the supervision weight λ (Appendix E.2), the update rule under the GNN seed (Appendix E.3), and the encoder backbone (Appendix E.4). Each ablation varies one hyperparameter and holds the others at the values selected for that benchmark (Table A.4); protocol and metric are as in Appendix A.4. Bold marks the best value in each row.

E.1 Unrolling depth T_{train}

Table E.1 varies T_{train} from 1 to 4 with T_{test} unchanged. Error decreases sharply from one to three unrolled steps and changes little from three to four, consistent with the trade-off of Section 4.2 between giving the encoder a useful horizon and keeping the gradient through the trajectory well conditioned.

Table E.1: Effect of the training unrolling depth T_{train} on test MSE ($\pm$ standard error over test graphs).

Dataset	$T_{\text{train}} = 1$	$T_{\text{train}} = 2$	$T_{\text{train}} = 3$	$T_{\text{train}} = 4$
IMDB	0.341 ± 0.198	0.214 ± 0.121	$\mathbf{0.085 \pm 0.080}$	0.112 ± 0.091
COLLAB	1.892 ± 0.412	0.841 ± 0.231	0.312 ± 0.089	$\mathbf{0.204 \pm 0.066}$
REDDIT	1.231 ± 0.312	0.612 ± 0.141	$\mathbf{0.165 \pm 0.018}$	0.198 ± 0.031
AIDS	0.312 ± 0.089	0.142 ± 0.041	$\mathbf{0.039 \pm 0.010}$	0.051 ± 0.014
DD	2.341 ± 0.412	1.102 ± 0.231	$\mathbf{0.582 \pm 0.067}$	0.631 ± 0.079
PTC-MM	0.412 ± 0.112	0.198 ± 0.061	$\mathbf{0.044 \pm 0.025}$	0.058 ± 0.031
MUTAG	0.512 ± 0.131	0.241 ± 0.071	$\mathbf{0.093 \pm 0.008}$	0.108 ± 0.011
ENZYMES	0.612 ± 0.141	0.312 ± 0.081	0.198 ± 0.051	$\mathbf{0.175 \pm 0.045}$
`C_random`	5.892 ± 0.812	4.231 ± 0.612	$\mathbf{2.791 \pm 0.229}$	2.912 ± 0.241
`RB`	2.341 ± 0.512	1.412 ± 0.312	$\mathbf{0.675 \pm 0.349}$	0.812 ± 0.381
`dsjc-style`	1.231 ± 0.312	0.712 ± 0.198	$\mathbf{0.270 \pm 0.089}$	0.312 ± 0.102
`brock-style`	2.892 ± 0.612	1.541 ± 0.341	$\mathbf{0.690 \pm 0.183}$	0.812 ± 0.201
`gen-style`	7.892 ± 1.102	5.231 ± 0.812	$\mathbf{2.942 \pm 0.239}$	3.102 ± 0.261
`p_hat-style`	3.892 ± 0.612	2.341 ± 0.412	$\mathbf{0.930 \pm 0.087}$	1.102 ± 0.098
`hamming-style`	18.231 ± 2.341	12.892 ± 1.712	$\mathbf{7.640 \pm 0.445}$	8.102 ± 0.512
`keller-style`	4.892 ± 0.612	3.102 ± 0.412	$\mathbf{1.620 \pm 0.134}$	1.812 ± 0.152

E.2 Supervision weight λ

Table E.2 varies λ in equation 4 from 0, the unsupervised Motzkin–Straus objective, to 1. The unsupervised variant is clearly worse, since the anchor is what supplies the target cardinality, but it does not lose feasibility: whether the output is a maximal clique is determined by the fixed-point structure of the dynamics Pelillo (1999); Bomze et al. (1999), not by the loss. On the synthetic families the unsupervised variant is on a par with some supervised solvers of Table 2, for instance 4.892 against GFnet's 4.820 on `dsjc-style`. The optimum lies at $\lambda \in \{0.5, 1\}$ on the real-world benchmarks and at $\lambda = 0.1$ on the synthetic families (Appendix A.3).

Table E.2: Effect of the supervision weight λ on test MSE ($\pm$ standard error over test graphs). $\lambda = 0$ is the unsupervised Motzkin–Straus objective.

Dataset	$\lambda = 0$	$\lambda = 0.1$	$\lambda = 0.5$	$\lambda = 1$
IMDB	4.231 ± 0.892	0.892 ± 0.198	0.241 ± 0.089	$\mathbf{0.085 \pm 0.080}$
COLLAB	18.341 ± 3.102	3.892 ± 0.712	0.892 ± 0.198	$\mathbf{0.204 \pm 0.066}$
REDDIT	12.892 ± 2.341	2.341 ± 0.512	$\mathbf{0.165 \pm 0.018}$	0.231 ± 0.041
AIDS	2.892 ± 0.512	0.412 ± 0.098	0.112 ± 0.031	$\mathbf{0.039 \pm 0.010}$
DD	8.231 ± 1.412	2.341 ± 0.412	0.892 ± 0.141	$\mathbf{0.582 \pm 0.067}$
PTC-MM	3.892 ± 0.712	0.512 ± 0.112	0.142 ± 0.051	$\mathbf{0.044 \pm 0.025}$
MUTAG	4.512 ± 0.892	0.712 ± 0.141	0.241 ± 0.061	$\mathbf{0.093 \pm 0.008}$
ENZYMES	1.892 ± 0.341	0.612 ± 0.121	$\mathbf{0.175 \pm 0.045}$	0.231 ± 0.058
`C_random`	18.231 ± 3.102	$\mathbf{2.791 \pm 0.229}$	3.412 ± 0.312	5.892 ± 0.712
`RB`	12.341 ± 2.102	$\mathbf{0.675 \pm 0.349}$	1.231 ± 0.312	2.892 ± 0.512
`dsjc-style`	4.892 ± 0.812	$\mathbf{0.270 \pm 0.089}$	0.412 ± 0.112	1.231 ± 0.241
`brock-style`	14.231 ± 2.341	$\mathbf{0.690 \pm 0.183}$	1.102 ± 0.231	3.412 ± 0.512
`gen-style`	16.892 ± 2.892	$\mathbf{2.942 \pm 0.239}$	4.231 ± 0.512	7.892 ± 1.102
`p_hat-style`	8.231 ± 1.412	$\mathbf{0.930 \pm 0.087}$	1.412 ± 0.198	3.231 ± 0.412
`hamming-style`	42.341 ± 6.892	$\mathbf{7.640 \pm 0.445}$	11.231 ± 1.341	18.892 ± 2.341
`keller-style`	9.892 ± 1.512	$\mathbf{1.620 \pm 0.134}$	2.341 ± 0.312	4.892 ± 0.712

E.3 Linear vs. exponential update under the GNN seed

The standalone-RD rows of Tables 1 and 2 show that the two updates are close when started from the barycentre; Table E.3 asks whether this remains true when they are started from $g_\theta(G)$. The

linear update is better on most benchmarks, and the exponential one is selected on validation for DD and MUTAG.

Table E.3: Linear vs. exponential replicator updates under the GNN seed (test MSE $\pm$ standard error over test graphs). On DD and MUTAG, where validation selects the exponential update, only that variant is reported (–).

Dataset	NeRD (Linear)	NeRD (Exponential)
IMDB	$\mathbf{0.085 \pm 0.080}$	0.098 ± 0.089
COLLAB	$\mathbf{0.204 \pm 0.066}$	0.231 ± 0.071
REDDIT	$\mathbf{0.165 \pm 0.018}$	0.189 ± 0.022
AIDS	$\mathbf{0.039 \pm 0.010}$	0.048 ± 0.013
DD	–	$\mathbf{0.582 \pm 0.067}$
PTC-MM	$\mathbf{0.044 \pm 0.025}$	0.061 ± 0.031
MUTAG	–	$\mathbf{0.093 \pm 0.008}$
ENZYMES	$\mathbf{0.175 \pm 0.045}$	0.201 ± 0.051
`C_random`	$\mathbf{2.791 \pm 0.229}$	3.102 ± 0.251
`RB`	$\mathbf{0.675 \pm 0.349}$	0.812 ± 0.381
`dsjc-style`	$\mathbf{0.270 \pm 0.089}$	0.312 ± 0.098
`brock-style`	$\mathbf{0.690 \pm 0.183}$	0.812 ± 0.201
`gen-style`	$\mathbf{2.942 \pm 0.239}$	3.231 ± 0.261
`p_hat-style`	0.930 ± 0.087	$\mathbf{0.910 \pm 0.083}$
`hamming-style`	$\mathbf{7.640 \pm 0.445}$	8.102 ± 0.512
`keller-style`	$\mathbf{1.620 \pm 0.134}$	1.790 ± 0.152

E.4 Encoder backbone

Table E.4 instantiates the convolution $\phi^{(\ell)}$ of the residual encoder (Section 4.1) with each of `GCN` Kipf & Welling (2016), `GAT` Velickovic et al. (2018), `GraphSAGE` Hamilton et al. (2017), and `SCTConv` Min et al. (2022), each trained with the full NeRD pipeline. GCN is best on fifteen of sixteen benchmarks and GraphSAGE on DD; GCN is used throughout.

Table E.4: Encoder backbone, with full NeRD training and inference (test MSE $\pm$ standard error over test graphs).

Dataset	GCN	GAT	GraphSAGE	SCTConv
IMDB	$\mathbf{0.085 \pm 0.080}$	0.198 ± 0.112	0.141 ± 0.098	0.512 ± 0.198
COLLAB	$\mathbf{0.204 \pm 0.066}$	0.512 ± 0.141	0.341 ± 0.112	1.892 ± 0.412
REDDIT	$\mathbf{0.165 \pm 0.018}$	0.412 ± 0.089	0.298 ± 0.071	1.341 ± 0.312
AIDS	$\mathbf{0.039 \pm 0.010}$	0.112 ± 0.031	0.081 ± 0.024	0.412 ± 0.098
DD	0.582 ± 0.067	1.231 ± 0.231	$\mathbf{0.521 \pm 0.058}$	2.892 ± 0.512
PTC-MM	$\mathbf{0.044 \pm 0.025}$	0.142 ± 0.051	0.098 ± 0.038	0.612 ± 0.141
MUTAG	$\mathbf{0.093 \pm 0.008}$	0.241 ± 0.058	0.178 ± 0.041	0.892 ± 0.198
ENZYMES	$\mathbf{0.175 \pm 0.045}$	0.312 ± 0.081	0.231 ± 0.061	0.712 ± 0.161
`C_random`	$\mathbf{2.791 \pm 0.229}$	4.892 ± 0.712	3.892 ± 0.512	12.341 ± 2.102
`RB`	$\mathbf{0.675 \pm 0.349}$	2.341 ± 0.512	1.412 ± 0.312	8.231 ± 1.412
`dsjc-style`	$\mathbf{0.270 \pm 0.089}$	0.892 ± 0.198	0.612 ± 0.141	4.231 ± 0.712
`brock-style`	$\mathbf{0.690 \pm 0.183}$	2.892 ± 0.512	0.812 ± 0.201	9.892 ± 1.712
`gen-style`	$\mathbf{2.942 \pm 0.239}$	6.892 ± 0.892	4.512 ± 0.612	18.231 ± 2.892
`p_hat-style`	$\mathbf{0.930 \pm 0.087}$	2.892 ± 0.412	1.892 ± 0.312	8.892 ± 1.341
`hamming-style`	$\mathbf{7.640 \pm 0.445}$	18.231 ± 2.341	12.892 ± 1.712	42.341 ± 5.892
`keller-style`	$\mathbf{1.620 \pm 0.134}$	4.892 ± 0.612	3.231 ± 0.412	12.892 ± 1.892

F Anytime lower bound

By equation 2, every iterate certifies $\omega(G) \geq 1/(1 - \mathbf{x}^{(t)\top}\mathbf{A}\mathbf{x}^{(t)})$. Table F.1 tracks this quantity along the NeRD trajectory on the six hardest synthetic families. The bound is loose at $t = 0$, where the seed still spreads mass over many vertices, reaches 79–96% of ω by $t = 50$, and at convergence is within 3% of ω on `RB`, `dsjc-style`, and `brock-style` and within 11% on `hamming-style`.

Two remarks keep this claim precise. First, the functional that the linear dynamics provably increase is the regularised f_α, not f; for $\alpha > 0$ the bound computed from f is valid at every t but not guaranteed to increase monotonically. An anytime certificate is therefore the running maximum $\max_{s \le t} 1/(1 - \mathbf{x}^{(s)\top}\mathbf{A}\mathbf{x}^{(s)})$, which is valid and non-decreasing by construction. Second, at a clique characteristic vector $\mathbf{x}^C$ the bound equals $|C|$ exactly, so at convergence it coincides with the size of the returned clique; its added value is before convergence, where it certifies a lower bound without a combinatorial check on the support.

Table F.1: Motzkin–Straus lower bound $1/(1 - \mathbf{x}^{(t)\top}\mathbf{A}\mathbf{x}^{(t)})$ along the NeRD trajectory, averaged over the test split. For reference, the second column gives the average clique number of the family (Table A.2).

Dataset	**Avg. ω (family)**	t=0	t=10	t=50	**Convergence**
`RB`	16.560	2.14	9.65	15.32	16.16
`dsjc-style`	17.582	3.82	11.45	16.89	17.38
`brock-style`	24.670	6.45	16.21	23.15	24.23
`gen-style`	34.484	13.56	23.84	31.45	33.28
`hamming-style`	19.112	6.12	9.85	15.12	17.01
`keller-style`	8.399	2.88	4.56	7.05	7.49

G Computational cost

G.1 Complexity

Table G.1 gives the cost of each stage in closed form, with L the number of message-passing layers and d the hidden width. Because $\mathbf{A}_\alpha\mathbf{x} = \mathbf{A}\mathbf{x} + \alpha\mathbf{x}$, a replicator step is a single sparse matrix–vector product and no $|V| \times |V|$ matrix needs to be formed.

Table G.1: Time and memory complexity of NeRD. The only term quadratic in any quantity is $|V|d^2$, from the dense layers of the encoder, which NeRD shares with every message-passing baseline. Structural features are computed once per graph and are not differentiated.

Stage	**Time**	**Memory**
Structural features (once per graph)	$O(\lvert E\rvert^{3/2})$	$O(\lvert E\rvert + \lvert V\rvert)$
Encoder	$O\big(L(\lvert E\rvert d + \lvert V\rvert d^2)\big)$	$O(L\lvert V\rvert d)$
One replicator step	$O(\lvert E\rvert)$	$O(\lvert E\rvert + \lvert V\rvert)$
Training (BPTT through T_{train} steps)	$O\big(L(\lvert E\rvert d + \lvert V\rvert d^2) + T_{\text{train}}\lvert E\rvert\big)$	$O(L\lvert V\rvert d + \lvert E\rvert + T_{\text{train}}\lvert V\rvert)$
Inference (T_{test} steps)	$O\big(L(\lvert E\rvert d + \lvert V\rvert d^2) + T_{\text{test}}\lvert E\rvert\big)$	$O(L\lvert V\rvert d + \lvert E\rvert)$

Three consequences follow. (i) The dynamics are linear in $|E|$: the T_{test} inference steps cost $O(T_{\text{test}}|E|)$ rather than $O(T_{\text{test}}|V|^2)$. For the largest DD graph ($|V| = 5{,}748$) a dense payoff would hold 3.3×10^7 entries, whereas the sparse adjacency holds at most $2|E| \approx 2.9 \times 10^4$ non-zeros. (ii) T_{train} enters memory only through the stored iterates, one scalar per node per step, so the short training horizon of Section 4.2 costs almost nothing in memory. (iii) At scale the binding constraint is the encoder: the concatenated representation $\mathbf{h}^{\text{cat}}$ holds $(L+1)d = 640$ values per node, about 2.6 GB in single precision for a graph with 10^6 nodes at inference, and a small multiple of that during training. The structural features require triangle counts, which we compute in $O(|E|^{3/2})$ time once per graph. Our benchmarks do not exceed $5{,}748$ vertices and we do not report measurements beyond that size.

G.2 Hardware and software

All methods run by us, NeRD and the re-run baselines, were trained and evaluated on the single machine of Table G.2 with the same software stack. This is a more modest setup than the multi-GPU servers used in some prior work, and the wall-clock numbers of Table G.3 should be read in that light.

Table G.2: Computational resources used for all experiments.

Component	Details
GPU	NVIDIA GeForce RTX 4090 (24 GB)
RAM	64 GB
CPU	Intel Xeon Silver 4210R @ 2.40 GHz
OS	Ubuntu 20.04.4 LTS
Python	3.10
PyTorch	2.3.0
PyTorch Geometric	latest stable release at the time of the experiments
Gurobi	10 (academic licence)

G.3 Wall-clock training and inference time

Table G.3 reports the average wall-clock time of one training epoch and the amortised inference time per graph, i.e. the cost of one forward pass with up to T_{test} replicator steps divided by the number of test graphs. Inference is dominated by the replicator steps. These measurements were taken with an implementation that stores the payoff as a dense matrix; the sparse step of Appendix G.1 computes the same iterates.

Table G.3: Wall-clock training time (seconds per epoch) and inference time (milliseconds per graph) of NeRD on the hardware of Table G.2.

Dataset	Train (s/ep)	Infer (ms/g)
IMDB	15.81	36.5
COLLAB	21.51	76.2
REDDIT	19.42	33.9
AIDS	8.78	5.4
DD	5.79	9.3
PTC-MM	9.14	5.9
MUTAG	50.41	21.8
ENZYMES	7.72	27.4
`C_random`	8.59	21.8
`RB`	10.58	29.3
`dsjc-style`	26.97	66.8
`brock-style`	21.15	49.9
`gen-style`	12.99	35.3
`p_hat-style`	24.72	60.5
`hamming-style`	10.30	27.9
`keller-style`	20.41	49.6

H Per-graph visualisations

This appendix shows the converged support of NeRD on two further instances beyond Figure 3, one from `RB` (Figure H.1) and one from IMDB (Figure H.2). Each figure is a 4×3 grid on a single instance. The first row shows the input graph with the ground-truth maximum clique in red, repeated across the three columns as a reference. The remaining rows show, in order, the continuous score of a GNN trained on equation 4 alone, the iterate of replicator dynamics from the barycentre, and the iterate of NeRD seeded with $g_\theta(G)$. The columns show, from left to right, the input graph as a positional reference, the soft scores at an intermediate iteration t near convergence, and the thresholded support read off as the predicted clique. Each row has its own colour bar, because the GNN-only output never collapses onto a characteristic vector and lives at lower per-vertex mass, while the replicator trajectories sharpen towards $1/k$ on a clique of size k. Within a row, the intermediate and converged panels share one scale. Instances are drawn at random from the

test split, and n, $\omega(G)$, the iteration t, and the size of the returned support are annotated below each panel.

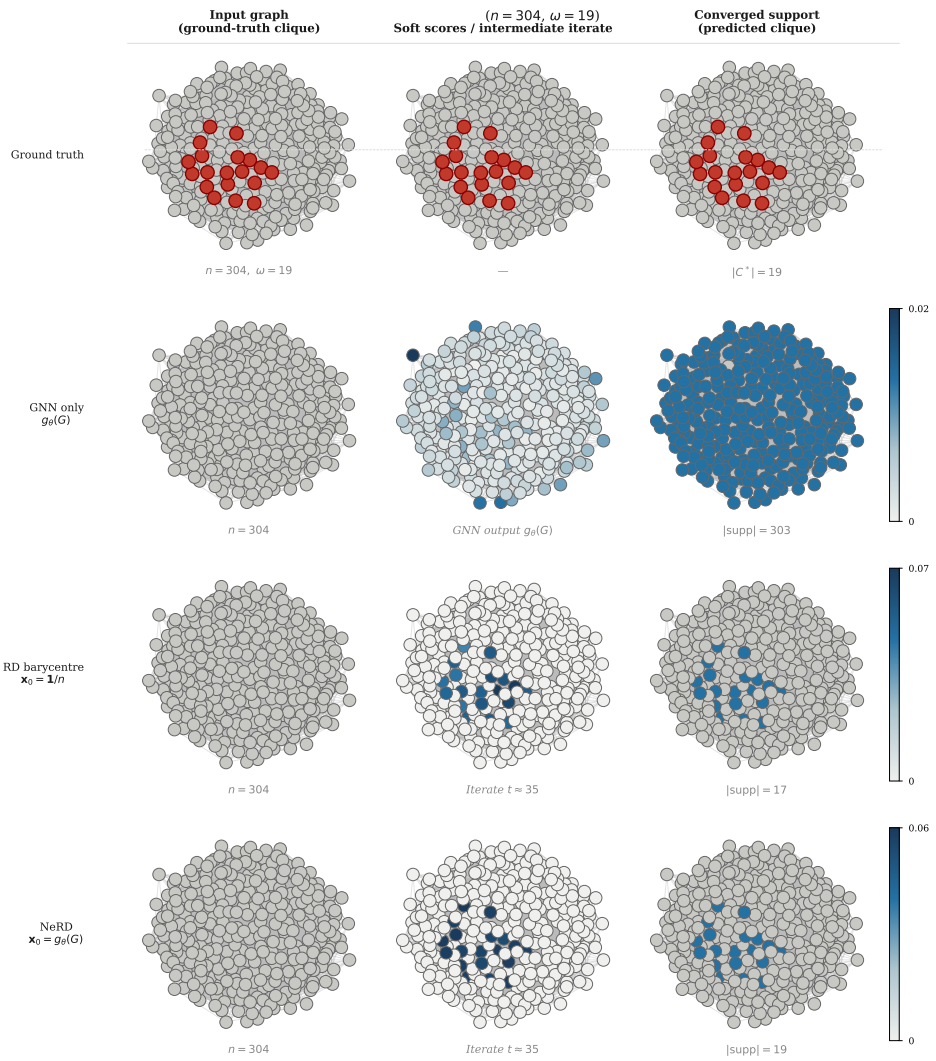


Figure H.1: Returned support of GNN-only, RD from the barycentre, and NeRD on a random test instance from RB.

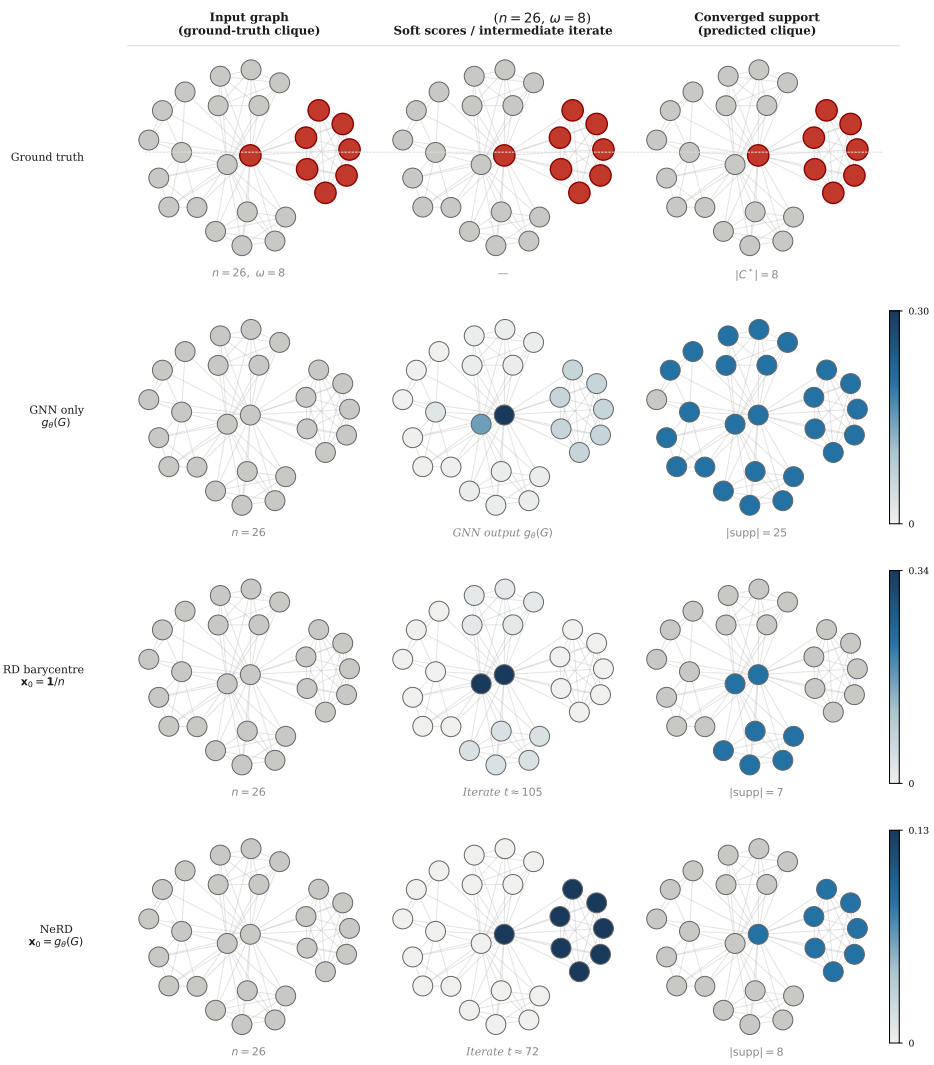


Figure H.2: Returned support of GNN-only, RD from the barycentre, and NeRD on a random test instance from IMDB.