LIFTED GEOMETRY-ALIGNED GNNS FOR MIN-k-PARTITION

Anonymous authors
Paper under double-blind review

ABSTRACT

Min-k-Partition is a fundamental NP-hard combinatorial optimization problem on graphs. Although learning methods have shown promise, generic architectures often fail to exploit the underlying problem geometry. This limitation can impair generalization and require costly adaptation for each new instance during inference. We introduce a KL family with node-specific weights for the probability simplex relaxation, accounting for node heterogeneity in the objective while respecting the simplex constraints. We select the node weights by minimizing a tractable spectral surrogate for relative smoothness, yielding the Geometry-Aligned KL (GA-KL) divergence. Mirror descent (MD) induced by GA-KL then provides an analytical solver aligned with the relaxed Min-k-Partition problem. We develop Lifted Geometry-Aligned Graph Neural Networks (LGA-GNNs) by lifting the GA-KL MD updates into latent spaces of higher dimension and adding trainable transformations to enhance representation ability. Recurrent application of the learned operator with shared and fixed parameters enables deeper refinement at inference without parameter updates or growth in model size. Experiments on random regular graphs with up to 10^6 nodes, Gset instances, and diverse real-world networks demonstrate strong solution quality and practical scalability under matched runtime budgets. Moreover, LGA-GNN generalizes effectively without retraining to graphs that are $100\times$ larger and to heterogeneous real-world structures.

1 INTRODUCTION

Min-k-Partition is one of the most fundamental problems in graph combinatorial optimization. It partitions the vertices of a weighted graph into k groups to minimize the total weight of edges whose endpoints lie in the same group (Ma & Hao, 2017). This objective naturally arises when assigning strongly connected vertex pairs to the same group is undesirable, such as frequency allocation and interference mitigation in wireless systems (Eisenblätter et al., 2002; Qiu et al., 2025a), as well as related partitioning formulations in chip design (Alpert et al., 1997) and statistical physics (Mézard et al., 1987; d'Auriac et al., 1997). Min-k-Partition is NP-hard, and the special case $k = 2$ is equivalent to the classical NP-complete Max-Cut problem (Karp, 2010).

Classical algorithms for Min-k-Partition include exact, approximation, and heuristic methods. Exact solvers based on semidefinite branch and cut methods provide optimality certificates, but in practice scale only to graphs with $\mathcal{O}(10^2)$ nodes (Ghaddar et al., 2011). Approximation algorithms typically start from semidefinite relaxations with randomized rounding, from the Goemans–Williamson method for $k = 2$ (Goemans & Williamson, 1995) to its extension for general k (Frieze & Jerrum, 1997). Later improvements use ϑ function relaxations (de Klerk et al., 2004) and SDP implementations with lower memory cost (Shinde et al., 2021). At larger scales, practical performance is dominated by heuristics, including continuous relaxations of low rank (Burer et al., 2002), tabu search (Kochenberger et al., 2013), local search (Benlic & Hao, 2013; Ma & Hao, 2017), and evolutionary frameworks (Wu & Hao, 2012; Goudet et al., 2024). Their computational cost nevertheless grows quickly with graph size, making high solution quality and practical efficiency difficult to achieve simultaneously.

To improve scalability, *learning to optimize* (L2O) trains parameterized optimizers over task distributions. By exploiting distributional priors, L2O can solve unseen instances in fewer iterations and achieve better solutions under comparable computational budgets (Bengio et al., 2021; Gasse et al., 2022; Chen et al., 2024). Early work often formulates graph partitioning as a sequential decision

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.

problem and trains a reinforcement learning policy that iteratively assigns vertices (Khalil et al., 2017; Barrett et al., 2020; 2022). Although effective at moderate scales, these methods require extensive rollouts, while their action spaces and sampling costs grow with graph size. A line of unsupervised methods instead directly optimizes the relaxed objective by optimizing neural parameters for each instance during inference, followed by rounding. GNN and hypergraph solvers instantiate this paradigm (Schuetz et al., 2022; Heydaribeni et al., 2024). Qiu et al. (2025b) further combine pretraining with adaptation for each instance. Such methods retain substantial inference latency and sensitivity to the adaptation budget, initialization, and hyperparameters. Their reliance on adaptation for each instance also suggests that generic neural architectures may not sufficiently exploit the problem geometry to transfer effective update rules across instances.

These limitations motivate architectures that more fully exploit problem geometry and transfer across instances without updating model parameters during inference. A useful route maps principled optimization updates to neural layers, combining explicit algorithmic structure with trainable transformations (Monga et al., 2021). Examples include LISTA for sparse coding (Gregor & LeCun, 2010), ADMM-Net for inverse problems (Yang et al., 2016a), PDHG-Net for linear programming (Li et al., 2024), GNNs obtained by unrolling primal-dual algorithms for constrained optimization (Hadou & Ribeiro, 2025), and data-driven mirror descent for learning Bregman geometries adapted to continuous convex optimization (Tan et al., 2023). Designs informed by geometry for large-scale combinatorial optimization on graphs, however, remain comparatively underexplored.

In this work, we propose Lifted Geometry-Aligned Graph Neural Networks (LGA-GNNs), a principled L2O framework for Min-k-Partition. We introduce a KL family with node-specific weights for the probability simplex relaxation, accounting for node heterogeneity in the objective while respecting the simplex constraints. We select the node weights by minimizing a tractable spectral surrogate for relative smoothness, yielding Geometry-Aligned KL (GA-KL) divergence. The resulting problem admits a closed-form normalized-degree solution for nonnegative graphs and an equivalent convex largest-eigenvalue formulation for mixed-sign graphs. Mirror descent (MD) induced by GA-KL then provides an analytical solver aligned with the relaxed Min-k-Partition problem. To enhance representation ability, we lift the GA-KL MD updates into latent spaces of higher dimension and introduce trainable transformations, resulting in LGA-GNNs. Recurrent parameter sharing enables progressive refinement by repeatedly applying the learned operator at inference with fixed parameters, without parameter updates or growth in model size.

Our main contributions are as follows:

- **Geometry-Aligned Divergence.** We formulate graph-dependent KL geometry selection as the minimization of a certified spectral surrogate for relative smoothness. The resulting problem admits an exact normalized-degree solution for nonnegative graphs and an equivalent convex largest-eigenvalue formulation for mixed-sign graphs.
- **Expressive Lifting.** Our lifted architecture extends the prescribed GA-KL MD update to a richer class of learnable update rules. We prove that a finite-width LGA-GNN can represent an output with an arbitrarily small residual under the GA-KL map.
- **Efficient Recurrent Refinement.** We recurrently reuse a shared learned operator with parameters fixed during inference. Increasing the refinement depth therefore requires neither parameter updates nor additional model parameters.
- **Comprehensive Empirical Evaluation.** Across random regular graphs, Gset instances, and diverse real-world networks, LGA-GNN delivers strong and consistent solution quality. Without retraining during inference, it generalizes to graphs that are $100\times$ larger than those encountered during training, scales to 10^6 nodes, and transfers effectively across heterogeneous real-world structures.

2 Preliminaries: Min-k-Partition Problem

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected weighted graph with $\mathcal{V} = \{1, \dots, N\}$ and a bounded symmetric adjacency matrix $\boldsymbol{W} \in \mathbb{R}^{N\times N}$. We exclude self-loops and isolated vertices, as they do not affect the partition objective and can be removed during preprocessing. For an assignment $\boldsymbol{r} \in \{1, \dots, k\}^N$,

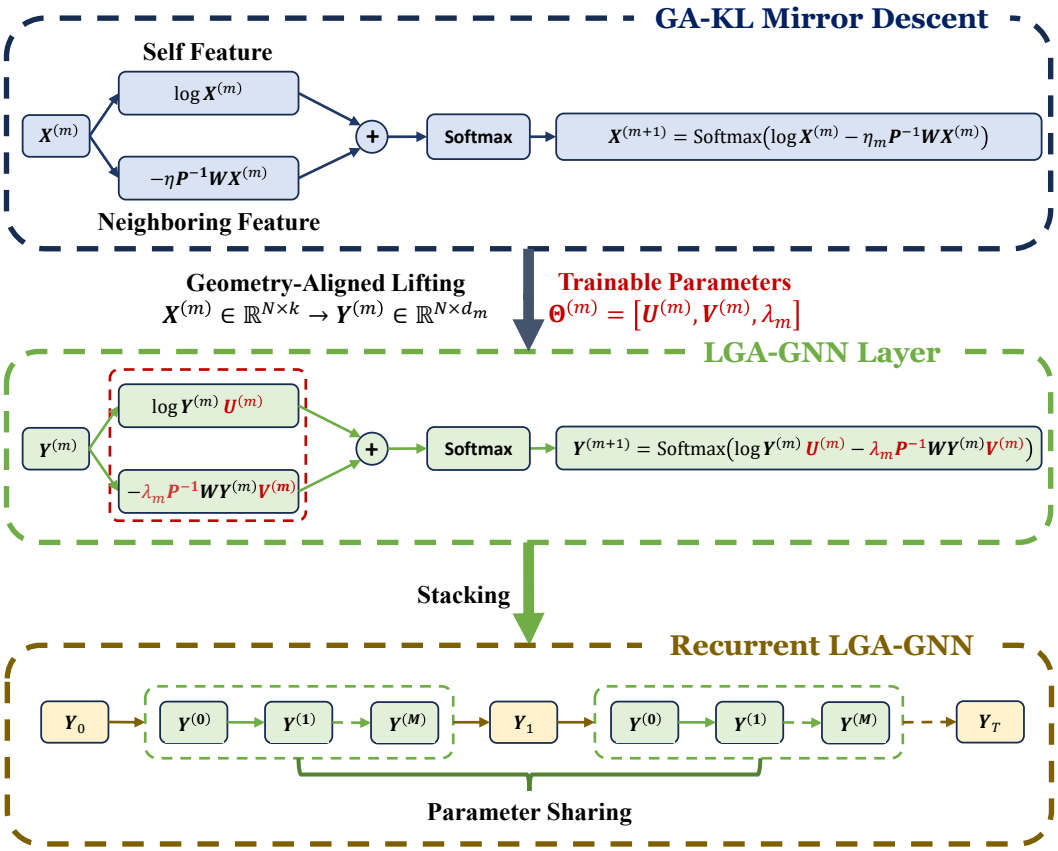


Figure 1: Overview of LGA-GNN. GA-KL mirror updates are lifted into trainable layers, which are stacked into a refinement operator reused with shared parameters across recurrent steps.

Min-k-Partition is

$$\min_{\boldsymbol{r}\in\{1,\dots,k\}^N} \quad \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} W_{ij}\,\mathbf{1}\{r_i = r_j\}.$$

Although this formulation does not explicitly require every label to be used (Ma & Hao, 2017), for nonnegative edge weights and $N \geq k$, enforcing nonempty groups does not change the optimal value.

Using one hot rows $\boldsymbol{X}_{i,:} = \boldsymbol{e}_{r_i}$ gives the equivalent quadratic form

$$\min_{\boldsymbol{X}} \quad f(\boldsymbol{X};\boldsymbol{W}) \coloneqq \frac{1}{2}\,\mathrm{Tr}(\boldsymbol{X}^\top \boldsymbol{W}\boldsymbol{X}), \qquad \text{s.t.} \quad \boldsymbol{X}_{i,:} \in \{\boldsymbol{e}_1,\dots,\boldsymbol{e}_k\},\ \forall i \in \mathcal{V}.$$

Let $\Delta_k \coloneqq \{\boldsymbol{x} \in \mathbb{R}^k_+ : \mathbf{1}^\top \boldsymbol{x} = 1\}$ and Δ_k^N denote the Cartesian product of N simplices. Replacing the one-hot constraint by its convex hull yields the probability simplex relaxation (Qiu et al., 2025b)

$$\min_{\boldsymbol{X}\in\Delta_k^N} f(\boldsymbol{X};\boldsymbol{W}). \tag{1}$$

The relaxation supports a simple sampling procedure for recovering discrete partitions.

Proposition 2.1 (Unbiased randomized rounding, Qiu et al., 2025b)**.** *For any $\boldsymbol{X} \in \Delta_k^N$, independently sample $\boldsymbol{Z}_{i,:} = \boldsymbol{e}_c$ with probability X_{ic}. Then*

$$\mathbb{E}[f(\boldsymbol{Z};\boldsymbol{W})] = f(\boldsymbol{X};\boldsymbol{W}).$$

Thus, optimizing (1) minimizes the expected objective of the sampled discrete solution. In practice, we draw multiple assignments and retain the one with the lowest objective (Qiu et al., 2025b).

3 GEOMETRY-ALIGNED KL DIVERGENCE FOR MIN-k-PARTITION

3.1 GEOMETRY-ALIGNED DIVERGENCE

A geometry for the relaxed problem (1) specifies how changes between feasible assignments are measured. Bregman divergences provide a general mechanism for encoding such geometry. Given a strictly convex differentiable generator ω, its associated Bregman divergence is

$$B_\omega(\boldsymbol{Y}, \boldsymbol{X}) = \omega(\boldsymbol{Y}) - \omega(\boldsymbol{X}) - \langle \nabla\omega(\boldsymbol{X}), \boldsymbol{Y} - \boldsymbol{X}\rangle$$

where different choices of ω induce different notions of proximity (Bregman, 1967; Beck, 2017). For Min-k-Partition, the generator should reflect two complementary structures:

- **Constraint geometry.** Each row of $\boldsymbol{X}$ is a probability distribution on the simplex.
- **Objective geometry.** The quadratic objective couples node variables through the adjacency matrix $\boldsymbol{W}$, and the strength of these interactions varies across nodes.

These structures motivate a node-wise weighted KL divergence. Specifically, given $\boldsymbol{p} \in \mathbb{R}^N_{++}$ satisfying $\mathbf{1}^\top \boldsymbol{p} = N$, define the weighted negative entropy and its induced Bregman divergence by

$$\omega(\boldsymbol{X}; \boldsymbol{p}) = \sum_{i=1}^N p_i \sum_{c=1}^k X_{ic} \log X_{ic}, \qquad B_\omega(\boldsymbol{Y}, \boldsymbol{X}; \boldsymbol{p}) = \sum_{i=1}^N p_i D_{\mathrm{KL}}(\boldsymbol{Y}_{i,:} \| \boldsymbol{X}_{i,:}). \tag{2}$$

Multiplying all entries of $\boldsymbol{p}$ by a common scalar rescales the divergence without changing its relative weighting across nodes. The unit-average constraint $\mathbf{1}^\top \boldsymbol{p} = N$ therefore selects a canonical representative, places all candidate geometries on a common scale, and makes $\boldsymbol{p} = \mathbf{1}$ the natural uniform reference (Qiu et al., 2025a).

We measure how well a weighted-KL geometry matches the objective through relative smoothness, which bounds the deviation of the objective from its first-order approximation.

Theorem 3.1 (Relative smoothness under weighted KL)**.** *Given $\boldsymbol{p} \in \mathbb{R}^N_{++}$ satisfying $\mathbf{1}^\top \boldsymbol{p} = N$, the objective function f is $L_{\boldsymbol{p}}$-smooth relative to the weighted KL divergence* (2) *on Δ_k^N. Specifically,*

$$f(\boldsymbol{Y}; \boldsymbol{W}) \le f(\boldsymbol{X}; \boldsymbol{W}) + \langle \nabla f(\boldsymbol{X}; \boldsymbol{W}), \boldsymbol{Y} - \boldsymbol{X}\rangle + L_{\boldsymbol{p}} B_\omega(\boldsymbol{Y}, \boldsymbol{X}; \boldsymbol{p}), \quad \forall \boldsymbol{X}, \boldsymbol{Y} \in \Delta_k^N, \tag{3}$$

where $\boldsymbol{P} = \mathrm{diag}(\boldsymbol{p})$ and $L_{\boldsymbol{p}} = \lambda_{\max}\big(\boldsymbol{P}^{-1/2} \boldsymbol{W} \boldsymbol{P}^{-1/2}\big)$.

The relative smoothness constant $L_{\boldsymbol{p}}$ provides a tractable spectral surrogate of relative smoothness. We select the geometry by minimizing this surrogate.

$$\min_{\boldsymbol{p} \in \mathbb{R}^N_{++}} \quad \lambda_{\max}\left(\boldsymbol{P}^{-1/2} \boldsymbol{W} \boldsymbol{P}^{-1/2}\right), \qquad \text{s.t.} \quad \mathbf{1}^\top \boldsymbol{p} = N. \tag{4}$$

The normalization fixes the overall scale of the geometry, so the optimization redistributes weight across nodes instead of enlarging all weights simultaneously.

For graphs with nonnegative edge weights, the geometry minimizing (4) has a closed-form solution.

Theorem 3.2 (Closed-form geometry for nonnegative graphs)**.** *Suppose that $W_{ij} \ge 0$ for all i, j. Let $d_i = \sum_{j=1}^N W_{ij}$, $\bar{d} = \frac{1}{N}\sum_{i=1}^N d_i$. Then, an optimal solution of* (4) *is $p_i^\star = \frac{d_i}{\bar{d}}$ and $L_{\boldsymbol{p}^\star} = \bar{d}$.*

When edge weights have mixed signs, signed degrees need not be positive and the closed form in Theorem 3.2 does not apply. The following result converts (4) directly into a convex largest-eigenvalue problem with a zero-sum diagonal shift.

Theorem 3.3 (Equivalent spectral formulation)**.** *Problem* (4) *is equivalent to*

$$\min_{\boldsymbol{q} \in \mathbb{R}^N} \quad \phi(\boldsymbol{q}) \coloneqq \lambda_{\max}\left(\boldsymbol{W} - \mathrm{diag}(\boldsymbol{q})\right), \qquad \textit{s.t.} \quad \mathbf{1}^\top \boldsymbol{q} = 0. \tag{5}$$

The two formulations have the same objective value under the maps $\boldsymbol{q} = t(\boldsymbol{p} - \mathbf{1}), \boldsymbol{p} = \mathbf{1} + \frac{\boldsymbol{q}}{t}$, where $t = L_{\boldsymbol{p}} = \phi(\boldsymbol{q})$. Consequently, if $\boldsymbol{q}^\star$ solves (5)*, then $t^\star = \phi(\boldsymbol{q}^\star), \boldsymbol{p}^\star = \mathbf{1} + \frac{\boldsymbol{q}^\star}{t^\star}$ solves* (4) *and satisfies $L_{\boldsymbol{p}^\star} = t^\star$.*

For solving problem (5), we propose a time-limited proximal spectral-bundle method, which is described in detail in Appendix A. In brief, leading eigenvectors generate affine cuts of the largest-eigenvalue function, and a small bundle of these cuts defines a stabilized model for updating $\boldsymbol{q}$. Each spectral query is computed by bounded restarted Lanczos iteration using sparse matrix–vector products. The method starts from $\boldsymbol{q} = \mathbf{0}$, which corresponds to the uniform geometry, and retains the best spectral objective encountered within the geometry budget. The selected node weights are then recovered through Theorem 3.3. Our theoretical analysis assumes exact spectral queries in Algorithm 1, and Appendix A.4 details the numerical implementation.

Corollary 3.4 (GA-KL nondegradation from uniform KL)**.** *Let $\boldsymbol{p}$ be the node weights returned by Algorithm 1. Instantiating* (2) *with $\boldsymbol{p}$ defines the* Geometry-Aligned KL *(GA-KL) divergence, whose certified relative-smoothness constant satisfies*

$$L_{\boldsymbol{p}} \le L_{\mathbf{1}} = \lambda_{\max}(\boldsymbol{W}).$$

Equality holds when the uniform geometry is retained, and the inequality is strict whenever the returned geometry has a smaller geometry-selection objective.

3.2 Mirror Descent Induced by GA-KL

Mirror descent minimizes a local linear approximation of the objective regularized by a Bregman divergence. Given the current iterate $\boldsymbol{X}^{(m)}$, the GA-KL geometry produces

$$\boldsymbol{X}^{(m+1)} = \arg\min_{\boldsymbol{Y} \in \Delta_k^N} \left\{ f(\boldsymbol{X}^{(m)}; \boldsymbol{W}) + \left\langle \nabla f(\boldsymbol{X}^{(m)}; \boldsymbol{W}), \boldsymbol{Y} - \boldsymbol{X}^{(m)} \right\rangle + \frac{1}{\eta} B_\omega(\boldsymbol{Y}, \boldsymbol{X}^{(m)}; \boldsymbol{p}) \right\}. \tag{6}$$

Euclidean and uniform KL divergences recover projected gradient descent and exponential gradient descent on the simplex, respectively (Kivinen & Warmuth, 1997; Beck & Teboulle, 2003). Substituting the selected GA-KL weights gives a similarly explicit update with node-adaptive scaling.

Proposition 3.5 (Closed form GA-KL mirror update)**.** *For a strictly positive $\boldsymbol{X}^{(m)} \in \Delta_k^N$, the solution of* (6) *is*

$$\boldsymbol{X}^{(m+1)} = \mathrm{Softmax}\Big(\log \boldsymbol{X}^{(m)} - \eta \boldsymbol{P}^{-1} \boldsymbol{W} \boldsymbol{X}^{(m)}\Big) =: \mathcal{T}(\boldsymbol{X}^{(m)}; \eta, \boldsymbol{p}), \tag{7}$$

where the logarithm is applied entrywise, and softmax is applied row-wise.

The non-degradation property of the selected geometry yields a common admissible step size, while the resulting descent bound preserves the improvement in $L_{\boldsymbol{p}}$.

Theorem 3.6 (Convergence rate of mirror descent induced by GA-KL)**.** *Let $\boldsymbol{p}$ be returned by Algorithm 1, and suppose $\|\boldsymbol{W}\|_{1,1} := \sum_{i,j=1}^N |W_{ij}| \le C_W$. Starting from any strictly positive $\boldsymbol{X}^{(0)} \in \Delta_k^N$, apply* (7) *with the constant step size $\eta_m \equiv (C_W + 1)^{-1}$. Then, after M iterations,*

$$\min_{0 \le m < M} B_\omega\Big(\mathcal{T}(\boldsymbol{X}^{(m)}; \eta_m, \boldsymbol{p}), \boldsymbol{X}^{(m)}; \boldsymbol{p}\Big) \le \frac{C_W}{M(C_W + 1 - L_{\boldsymbol{p}})}.$$

Remark 3.7 (Nondegradation of the convergence guarantee)**.** Under the same constant step size, the finite-time geometry returned by Algorithm 1 has a convergence guarantee no worse than that of uniform KL, with a strict improvement whenever $L_{\boldsymbol{p}} < L_{\mathbf{1}}$.

Although the GA-KL mirror update is aligned with the geometry of the relaxed problem, it remains a fixed analytical iteration in the original k-dimensional assignment space. Its representational capacity is therefore restricted to a prescribed family of trajectories. We next lift the simplex states into higher-dimensional latent spaces and parameterize both the mirror transformation and the graph-interaction transformation. Complete proofs of the theoretical results in this section are collected in Appendix C.

4 Lifted Geometry-Aligned GNNs

4.1 Design of Lifted Geometry-Aligned GNNs

Each GA-KL mirror update in (7) consists of three steps:

(i) **Mirror map:** applying $\log(\boldsymbol{X}^{(m)})$ to map simplex variables into dual coordinates.

(ii) **Gradient step with geometry scaling:** incorporating $-\eta \boldsymbol{P}^{-1}\boldsymbol{W}\boldsymbol{X}^{(m)}$.

(iii) **Softmax retraction:** mapping the result back to Δ_k^N by applying softmax to each row.

Each row of $\boldsymbol{X}^{(m)}$ is a probability vector over k partitions and can be interpreted as the feature representation of one node. The product $\boldsymbol{W}\boldsymbol{X}^{(m)}$ aggregates these features from neighboring nodes, $\boldsymbol{P}^{-1}$ adjusts the interaction scale for each node according to the selected geometry, and the logarithmic mirror map and softmax retraction preserve the simplex structure. This message-passing interpretation motivates our learnable graph architecture.

To enhance representation ability while retaining this structure, we lift the node states to higher-dimensional latent embeddings. Concretely, we generalize

$$\boldsymbol{X}^{(m)} \in \Delta_k^N \quad \Longrightarrow \quad \boldsymbol{Y}^{(m)} \in \Delta_{d_m}^N, \qquad d_m \geq k.$$

The mirror map and graph interaction are extended jointly as

$$\log(\boldsymbol{X}^{(m)}) \implies \log(\boldsymbol{Y}^{(m)})\boldsymbol{U}^{(m)}, \qquad \boldsymbol{P}^{-1}\boldsymbol{W}\boldsymbol{X}^{(m)} \implies \boldsymbol{P}^{-1}\boldsymbol{W}\boldsymbol{Y}^{(m)}\boldsymbol{V}^{(m)},$$

where $\boldsymbol{U}^{(m)}$ maps the dual variables to the next latent dimension and $\boldsymbol{V}^{(m)}$ adapts the graph-based message passing to the underlying data distribution. We retain softmax to ensure feasibility in the next probability simplex.

Accordingly, each lifted layer generalizes one GA-KL iteration through trainable transformations:

$$\boldsymbol{Y}^{(m+1)} = \mathrm{Softmax}\Big(\log(\boldsymbol{Y}^{(m)})\boldsymbol{U}^{(m)} - \lambda_m \boldsymbol{P}^{-1}\boldsymbol{W}\boldsymbol{Y}^{(m)}\boldsymbol{V}^{(m)}\Big), \tag{8}$$

where $\boldsymbol{U}^{(m)}, \boldsymbol{V}^{(m)} \in \mathbb{R}^{d_m \times d_{m+1}}$ and $\lambda_m \in \mathbb{R}$ are trainable, with $d_0 = d_M = k$ and internal widths $d_m \geq k$. Stacking M such layers yields an LGA-GNN. The trainable transformations enlarge the update family while the architecture retains an explicit realization of the GA-KL updates.

Theorem 4.1 (Layerwise alignment with GA-KL updates)**.** *Fix the selected geometry $\boldsymbol{p} \in \mathbb{R}_{++}^N$, and let $\{\boldsymbol{X}^{(m)}\}_{m=0}^M$ be any finite GA-KL mirror trajectory from a strictly positive initialization. For every $\varepsilon_{\mathrm{align}} > 0$ and any width sequence satisfying $d_0 = d_M = k$ and $d_m \geq k$, there exists an LGA-GNN with M layers such that*

$$\Big\| \boldsymbol{Y}^{(m)} - \begin{bmatrix} \boldsymbol{X}^{(m)} & \boldsymbol{0}_{N\times(d_m-k)} \end{bmatrix} \Big\|_F^2 \leq \varepsilon_{\mathrm{align}}, \qquad m = 0, \ldots, M. \tag{9}$$

Theorem 4.1 establishes representation inclusion: lifting preserves finite analytical GA-KL trajectories while allowing additional learnable transformations. Combining this inclusion with the finite-iteration GA-KL guarantee shows that LGA-GNN can represent an output with a small residual under the analytical GA-KL map, characterizing the representational capacity of LGA-GNN.

Theorem 4.2 (Representation of GA-KL near-stationarity via normalized outputs)**.** *Let $\boldsymbol{p}$ be returned by Algorithm 1, suppose $\|\boldsymbol{W}\|_{1,1} \leq C_W$, and initialize the GA-KL trajectory at any strictly positive $\boldsymbol{X}^{(0)} \in \Delta_k^N$. Fix a width bound $d_{\max} \geq k$. Then, for every $\varepsilon_{\mathrm{stat}} > 0$, there exists an LGA-GNN initialized at $\boldsymbol{Y}^{(0)} = \boldsymbol{X}^{(0)}$, with depth $M = \lceil 2C_W/\varepsilon_{\mathrm{stat}} \rceil$, $d_0 = d_M = k$, and $k \leq d_m \leq d_{\max}$ for $0 < m < M$, such that at least one normalized layer output $\widehat{\boldsymbol{Y}}^{(m)} \in \Delta_k^N$, $0 \leq m < M$, satisfies*

$$B_\omega\Big(\mathcal{T}(\widehat{\boldsymbol{Y}}^{(m)}; \eta, \boldsymbol{p}), \widehat{\boldsymbol{Y}}^{(m)}; \boldsymbol{p}\Big) \leq \varepsilon_{\mathrm{stat}}, \qquad \eta = (C_W + 1)^{-1}. \tag{10}$$

Here, the first k channels are normalized rowwise as $\widehat{Y}_{ic}^{(m)} = \frac{Y_{ic}^{(m)}}{\sum_{j=1}^k Y_{ij}^{(m)}}$ for $i \in [N], c \in [k]$.

4.2 Recurrent Refinement via Parameter Sharing

The LGA-GNN with M layers defines a recurrent refinement operator $\mathcal{F}_{\boldsymbol{\Theta}}$ that maps a relaxed solution in Δ_k^N to a refined solution. Given Gaussian logits $\boldsymbol{H} \in \mathbb{R}^{N\times k}$ scaled using Xavier initialization (Glorot & Bengio, 2010), we initialize $\boldsymbol{Y}_0 = \mathrm{Softmax}(\boldsymbol{H}) \in \Delta_k^N$ and recursively apply the same operator,

$$\boldsymbol{Y}_{t+1} = \mathcal{F}_{\boldsymbol{\Theta}}(\boldsymbol{Y}_t; \boldsymbol{W}, \boldsymbol{p}), \qquad t = 0, 1, \ldots, T-1.$$

The parameters of all M layers are reused across refinement steps. Parameter sharing therefore extends the effective search horizon without increasing the parameter count. Because learned recurrent updates need not retain the monotonicity of analytical mirror descent, inference tracks and returns the best observed relaxed solution.

4.3 Network Training and Inference

During training, we use the multi-step objective

$$\min_{\boldsymbol{\Theta}} \mathcal{L}_{\boldsymbol{\Theta}} \coloneqq \sum_{\boldsymbol{W}\in\mathcal{D}} \sum_{t=1}^{T_{\text{train}}} f(\boldsymbol{Y}_t;\boldsymbol{W}), \quad \text{s.t. } \boldsymbol{Y}_{t+1} = \mathcal{F}_{\boldsymbol{\Theta}}(\boldsymbol{Y}_t;\boldsymbol{W},\boldsymbol{p}),\ t=0,\dots,T_{\text{train}}-1. \tag{11}$$

A fixed horizon encourages the learned operator to remain useful under repeated composition. In practice, we optimize a detached-state surrogate of (11) for memory-efficient and stable recurrent training: at each refinement step t, we treat $\boldsymbol{Y}_{t-1}$ as constant, differentiate only the current loss $f(\mathcal{F}_{\boldsymbol{\Theta}}(Y_{t-1};\boldsymbol{W},\boldsymbol{p});\boldsymbol{W})$, immediately take one optimizer step on Θ, and detach Y_t before continuing to the next step.

Remark 4.3 (Empirical risk containment)**.** For a finite training set and recurrent horizon, let $\mathcal{L}_{\text{GAKL}}$ denote the empirical objective produced by the analytical GA-KL updates under the same initialization and a common step size. The identity blocks and step-size parameters in the aligned construction are independent of the recurrent index, while finiteness of the training set and horizon permits the channel-suppression parameters to be chosen uniformly over all recurrent occurrences. Consequently, for every $\delta > 0$, there exists a parameter-shared recurrent LGA-GNN satisfying

$$\mathcal{L}_{\boldsymbol{\Theta}} \le \mathcal{L}_{\text{GAKL}} + \delta, \qquad \text{and hence} \qquad \inf_{\boldsymbol{\Theta}} \mathcal{L}_{\boldsymbol{\Theta}} \le \mathcal{L}_{\text{GAKL}}. \tag{12}$$

During inference, we freeze $\boldsymbol{\Theta}$ and apply the operator for T_{infer} steps. Because the learned recurrent updates may be nonmonotonic, we track the best observed relaxed solution over the entire trajectory:

$$t^* = \arg\min_{0\le t\le T_{\text{infer}}} f(\boldsymbol{Y}_t;\boldsymbol{W}), \qquad \boldsymbol{Y}^* = \boldsymbol{Y}_{t^*}.$$

The complete training and inference procedure is summarized in Appendix B. Proofs and supporting constructions for the representation results in this section are collected in Appendix D.

5 Numerical Evaluation

5.1 Experimental Setup

We evaluate `LGA-GNN` against a diverse set of classical and L2O baselines for Min-k-Partition. Appendix B shows the implementation details of `LGA-GNN` and runtime measurements.

Baseline Algorithms. We compare our method against the following baselines: (i) `GW-Manopt` (Boumal et al., 2016), `GW-SketchyCGAL` (Yurtsever et al., 2021), and `GW-HALLaR` (Monteiro et al., 2026): three scalable methods instantiated on the Goemans–Williamson SDP relaxation (Goemans & Williamson, 1995), used for $k = 2$. (ii) `Genetic` (Li & Wang, 2016): a genetic algorithm for Min-k-Partition. (iii) `Rank-2` (Burer et al., 2002): a Burer–Monteiro factorization-based SDP solver with rank 2 for $k = 2$. (iv) `BLS` (Benlic & Hao, 2013): a breakout local search heuristic for signed $k = 2$ instances; (v) `Gurobi` (Gurobi Optimization, LLC, 2026): a classical MIQP solver for the discrete problem. (vi) `ECO-DQN` (Barrett et al., 2020): an RL approach combining Q-learning with test-time exploratory refinement, applicable to $k = 2$. (vii) `PI-GNN` (Schuetz et al., 2022): an unsupervised GNN solver for QUBO, applicable to $k = 2$. (viii) `HypOp` (Heydaribeni et al., 2024): an unsupervised hypergraph neural solver applicable to $k = 2$. (ix) `ROS` (Qiu et al., 2025b): a relax-optimize-and-sample framework based on unsupervised GNNs for Min-k-Partition.

Testing Datasets. We evaluate all methods on the following benchmarks. All experiments are repeated over 10 independent runs.

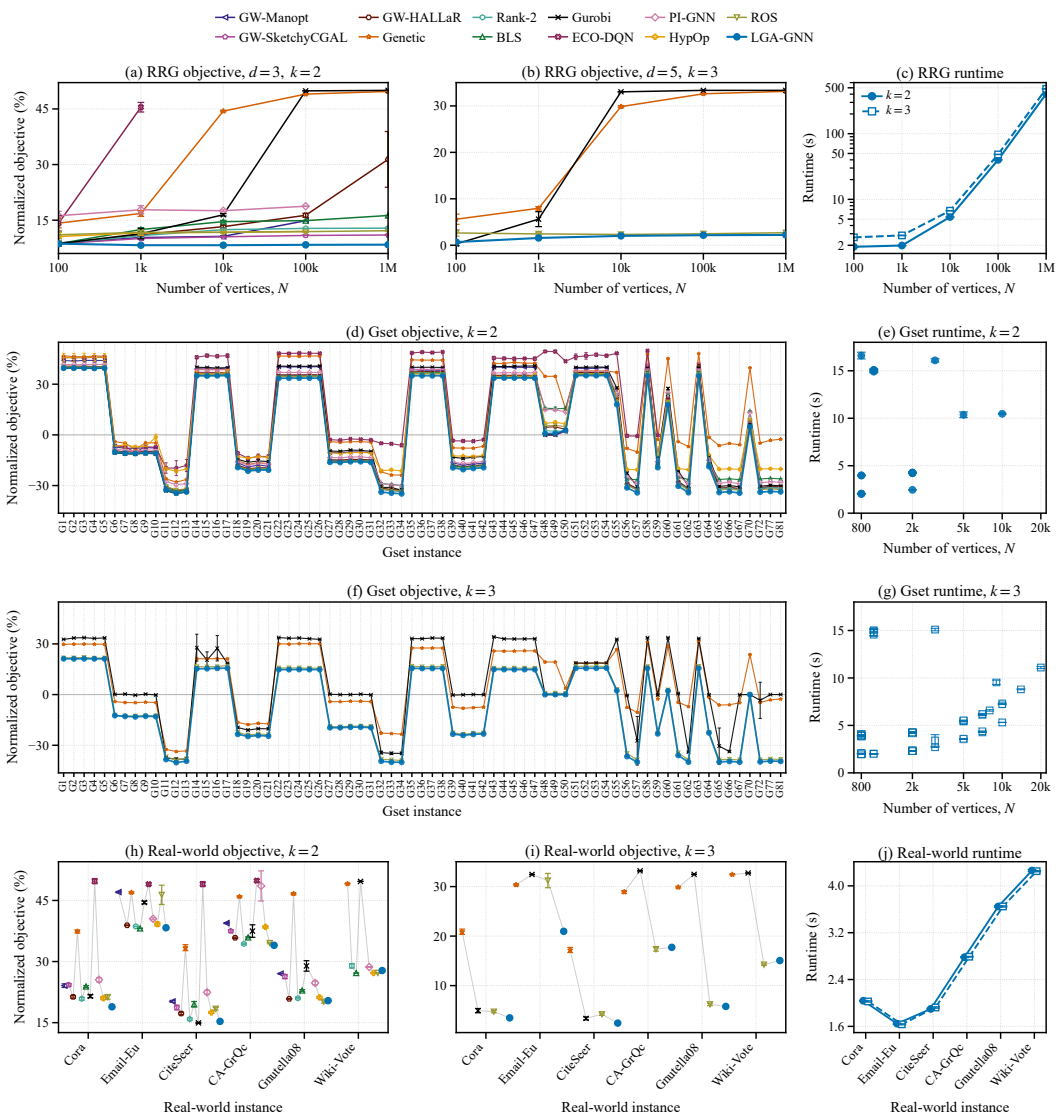


Figure 2: Evaluation results on RRG, Gset, and real-world datasets. Each point represents the mean of 10 independent runs, and the error bars indicate one standard deviation. Lower normalized objectives are better. Baselines are evaluated under wall-clock limits set by the end-to-end runtime of LGA-GNN with $T_{\text{infer}} = 100$.

- **Random regular graphs (RRGs)** (Schuetz et al., 2022): Each node has degree r. We use 3-regular graphs for $k = 2$ and 5-regular graphs for $k = 3$, with graph sizes $N \in \{10^2, 10^3, 10^4, 10^5, 10^6\}$.
- **Gset** (Ye, 2003): A standard graph benchmark suite containing toroidal, planar, and random graphs. We evaluate all 71 instances with N ranging from 800 to 20,000. Their edge densities range from approximately 0.02% to 6%, with edge weights in $\{-1, +1\}$.
- **Real-world datasets** (Leskovec & Krevl, 2014; Yang et al., 2016b): We consider two academic citation networks (Cora and CiteSeer), an email communication network (Email-Eu-core), a coauthorship network (CA-GrQc), a Gnutella peer-to-peer network (p2p-Gnutella08), and a Wikipedia administrator-election voting network (Wiki-Vote). All datasets are converted into undirected simple graphs by removing self-loops and duplicate edges. For Cora and CiteSeer, we retain the largest connected component. The resulting graphs range from $N = 986$ to 7,066 nodes and contain between 3,679 and 100,736 edges.

5.2 Main Results

Figure 2 summarizes the solution quality and runtime of LGA−GNN at $T_{\text{infer}} = 100$. Panels (a)-(c) report the objectives for $k = 2$ and $k = 3$ and the runtime on RRGs as functions of N. Panels (d) and (f) report the objectives across the 71 Gset instances for $k = 2$ and $k = 3$, while panels (e) and (g) plot the corresponding runtime against graph size. Panels (h)-(j) report the two objectives and runtime across the real-world instances. We normalize the partition objective as $100 \times \text{Obj}/|\mathcal{E}|$, where lower values are better. For each instance and k, every baseline is assigned the end-to-end runtime of LGA−GNN as its wall-clock limit. The runtime panels therefore show only LGA−GNN, which defines the common computational budget. Per-instance objective–time trajectories for all baselines on RRG, Gset, and real-world datasets are provided in Appendix F.

RRGs. LGA−GNN achieves the best or tied-best objective in 9 of the 10 RRG settings and is consistently best for both $k = 2$ and $k = 3$ when $N \geq 10^3$. For $k = 2$, it reduces the normalized objective relative to the strongest baseline by 1.81 percentage points at $N = 10^3$ and 2.60 at $N = 10^6$, while its objective remains nearly unchanged as the graph grows. Although trained only on graphs with at most 10^4 nodes, LGA−GNN generalizes directly to graphs that are $100\times$ larger, processing the 10^6-node instances in approximately 404 and 482 seconds for $k = 2$ and $k = 3$, respectively.

Gset. Across the 142 Gset instance-k settings, LGA−GNN obtains the best objective in 125 settings and ties for the best in one. It ranks first on 57 of the 71 instances for $k = 2$ and 68 of the 71 instances for $k = 3$, improving upon the strongest competing method by an average of 0.60 and 0.87 percentage points, respectively. Notably, LGA−GNN outperforms ROS on every Gset instance for both values of k, establishing a consistent advantage across diverse graph sizes, densities, and structures.

Real-World Graphs. LGA−GNN achieves the best objective in 6 of the 12 real-world settings, including both values of k on Cora, after training exclusively on synthetic random geometric graphs. Relative to the strongest baseline on each instance, it improves the objective by an average of 0.16 percentage points for $k = 2$ and 1.80 for $k = 3$, with an improvement of 9.39 percentage points on Email-Eu-core for $k = 3$. These results demonstrate effective transfer to heterogeneous citation, communication, collaboration, peer-to-peer, and social networks, with only 1.6–4.3 seconds required for 100 inference steps.

We conduct comprehensive ablation studies in Appendix E. A paired ablation in Appendix E.1 isolates the contribution of the spectral-bundle geometry on the signed Gset instances and shows consistent improvements over uniform KL. Appendix E.2 further compares LGA−GNN with a matched pretrained GNN under the same training data, architecture scale, and inference schedule, isolating the contribution of the GA-KL design. Further ablations in Appendices E.3-E.7 show that learned depth substantially improves over analytical GA-KL iterations, performance saturates at moderate width, and recurrent refinement provides a smooth quality–runtime trade-off.

6 Conclusion

We introduced LGA-GNN, a scalable L2O framework for Min-k-Partition. GA-KL aligns the divergence with the simplex constraints and graph-dependent node interactions through a principled spectral geometry-selection criterion. It admits principled weight selection and induces a closed-form mirror update with a finite-iteration convergence guarantee. Lifting this update into trainable latent spaces improves representation ability. Offline training learns a refinement operator that is reused through recurrent parameter sharing, avoiding per-instance parameter optimization at inference. The resulting solver achieves strong solution quality and practical scalability across random regular graphs, Gset instances, and diverse real-world networks. Without retraining, it generalizes to graphs that are $100\times$ larger than those encountered during training, scales to 10^6 nodes, and transfers effectively to heterogeneous real-world structures. More broadly, the results suggest that learned graph optimizers benefit from coupling geometry aligned with the optimization problem and latent representations that expand the analytical updates.

AI USE STATEMENT

We used large language models to assist with drafting and revising portions of the manuscript, improving readability, and checking consistency across the text, equations, and reported results. This assistance included refining the wording of mathematical statements and their proofs. The authors reviewed and revised all AI-assisted text and checked the mathematical arguments and empirical claims against the underlying derivations and experimental results. The authors take full responsibility for the final content of the paper, including all claims, proofs, and reported results.

REFERENCES

Charles J Alpert, Jen-Hsin Huang, and Andrew B Kahng. Multilevel circuit partitioning. In *Proceedings of the 34th annual Design Automation Conference*, pp. 530–533, 1997.

Thomas Barrett, William Clements, Jakob Foerster, and Alex Lvovsky. Exploratory combinatorial optimization with reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 3243–3250, 2020.

Thomas D Barrett, Christopher WF Parsonson, and Alexandre Laterre. Learning to solve combinatorial graph partitioning problems via efficient exploration. *arXiv preprint arXiv:2205.14105*, 2022.

Amir Beck. *First-order methods in optimization*. SIAM, 2017.

Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, 2021.

Una Benlic and Jin-Kao Hao. Breakout local search for the max-cut problem. *Engineering Applications of Artificial Intelligence*, 26(3):1162–1173, 2013.

Nicolas Boumal, Vlad Voroninski, and Afonso Bandeira. The non-convex burer-monteiro approach works on smooth semidefinite programs. *Advances in Neural Information Processing Systems*, 29, 2016.

Lev M Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR computational mathematics and mathematical physics*, 7(3):200–217, 1967.

Samuel Burer, Renato DC Monteiro, and Yin Zhang. Rank-two relaxation heuristics for max-cut and other binary quadratic programs. *SIAM Journal on Optimization*, 12(2):503–521, 2002.

Xiaohan Chen, Jialin Liu, and Wotao Yin. Learning to optimize: A tutorial for continuous and mixed-integer optimization. *Science China Mathematics*, pp. 1–72, 2024.

JC Anglès d'Auriac, M Preissmann, and A Sebö. Optimal cuts in graphs and statistical mechanics. *Mathematical and Computer Modelling*, 26(8-10):1–11, 1997.

Etienne de Klerk, Dmitrii V Pasechnik, and Joost P Warners. On approximate graph colouring and max-k-cut algorithms based on the θ-function. *Journal of Combinatorial Optimization*, 8:267–294, 2004.

Andreas Eisenblätter, Martin Grötschel, and Arie Koster. Frequency planning and ramifications of coloring. *Discussiones Mathematicae Graph Theory*, 22(1):51–88, 2002.

Alan Frieze and Mark Jerrum. Improved approximation algorithms for max k-cut and max bisection. *Algorithmica*, 18(1):67–81, 1997.

Maxime Gasse, Simon Bowly, Quentin Cappart, Jonas Charfreitag, Laurent Charlin, Didier Chételat, Antonia Chmiela, Justin Dumouchelle, Ambros Gleixner, Aleksandr M Kazachkov, et al. The machine learning for combinatorial optimization competition (ml4co): Results and insights. In *NeurIPS 2021 competitions and demonstrations track*, pp. 220–231. PMLR, 2022.

Bissan Ghaddar, Miguel F Anjos, and Frauke Liers. A branch-and-cut algorithm based on semidefinite programming for the minimum k-partition problem. *Annals of Operations Research*, 188(1):155–174, 2011.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6): 1115–1145, 1995.

Olivier Goudet, Adrien Goëffon, and Jin-Kao Hao. A large population island framework for the unconstrained binary quadratic problem. *Computers & Operations Research*, 168:106684, 2024.

Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th international conference on international conference on machine learning*, pp. 399–406, 2010.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2026. URL `https://www.gurobi.com`.

Samar Hadou and Alejandro Ribeiro. Unrolled graph neural networks for constrained optimization, 2025. URL `https://arxiv.org/abs/2509.17156`.

Nasimeh Heydaribeni, Xinrui Zhan, Ruisi Zhang, Tina Eliassi-Rad, and Farinaz Koushanfar. Distributed constrained combinatorial optimization leveraging hypergraph neural networks. *Nature Machine Intelligence*, 6(6):664–672, 2024.

Richard M Karp. *Reducibility among combinatorial problems*. Springer, 2010.

Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. *Advances in neural information processing systems*, 30, 2017.

Jyrki Kivinen and Manfred K Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and computation*, 132(1):1–63, 1997.

Gary A Kochenberger, Jin-Kao Hao, Zhipeng Lü, Haibo Wang, and Fred Glover. Solving large scale max cut problems via tabu search. *Journal of Heuristics*, 19(4):565–571, 2013.

Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

Bingheng Li, Linxin Yang, Yupeng Chen, Senmiao Wang, Haitao Mao, Qian Chen, Yao Ma, Akang Wang, Tian Ding, Jiliang Tang, and Ruoyu Sun. PDHG-unrolled learning-to-optimize method for large-scale linear programming. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 29164–29180. PMLR, 21–27 Jul 2024.

Panxing Li and Jing Wang. Pci planning method based on genetic algorithm in lte network. *Telecommunications Science*, 32(3):2016082, 2016.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Fuda Ma and Jin-Kao Hao. A multiple search operator heuristic for the max-k-cut problem. *Annals of Operations Research*, 248:365–403, 2017.

Marc Mézard, Giorgio Parisi, and Miguel Angel Virasoro. *Spin glass theory and beyond: An Introduction to the Replica Method and Its Applications*, volume 9. World Scientific Publishing Company, 1987.

Vishal Monga, Yuelong Li, and Yonina C Eldar. Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing. *IEEE Signal Processing Magazine*, 38(2):18–44, 2021.

Renato DC Monteiro, Arnesh Sujanani, and Diego Cifuentes. A low-rank augmented lagrangian method for large-scale semidefinite programming based on a hybrid convex-nonconvex approach. *Mathematical Programming*, pp. 1–57, 2026.

Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 4602–4609, 2019.

Beresford N Parlett. *The symmetric eigenvalue problem*. SIAM, 1998.

Yeqing Qiu, Chengpiao Huang, Ye Xue, Zhipeng Jiang, Qingjiang Shi, Dong Zhang, and Zhi-Quan Luo. Relaxation-free min-k-partition for PCI assignment in 5G networks. *IEEE Transactions on Signal Processing*, 73:3931–3946, 2025a. doi: 10.1109/TSP.2025.3604409.

Yeqing Qiu, Ye Xue, Akang Wang, Yiheng Wang, Qingjiang Shi, and Zhi-Quan Luo. ROS: A GNN-based relax-optimize-and-sample framework for max-k-cut problems. In *Forty-second International Conference on Machine Learning*, 2025b.

Martin JA Schuetz, J Kyle Brubaker, and Helmut G Katzgraber. Combinatorial optimization with physics-inspired graph neural networks. *Nature Machine Intelligence*, 4(4):367–377, 2022.

Nimita Shinde, Vishnu Narayanan, and James Saunderson. Memory-efficient approximation algorithms for max-k-cut and correlation clustering. *Advances in Neural Information Processing Systems*, 34:8269–8281, 2021.

Hong Ye Tan, Subhadip Mukherjee, Junqi Tang, and Carola-Bibiane Schönlieb. Data-driven mirror descent with input-convex neural networks. *SIAM Journal on Mathematics of Data Science*, 5(2): 558–587, 2023.

Qinghua Wu and Jin-Kao Hao. A memetic approach for the max-cut problem. In *International Conference on Parallel Problem Solving from Nature*, pp. 297–306. Springer, 2012.

Yan Yang, Jian Sun, Huibin Li, and Zongben Xu. Deep admm-net for compressive sensing mri. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016a. URL https://proceedings.neurips.cc/paper_files/paper/2016/file/1679091c5a880faf6fb5e6087eb1b2dc-Paper.pdf.

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pp. 40–48. PMLR, 2016b.

Yinyu Ye. The gset dataset. https://web.stanford.edu/~yyye/yyye/Gset/, 2003.

Alp Yurtsever, Joel A Tropp, Olivier Fercoq, Madeleine Udell, and Volkan Cevher. Scalable semidefinite programming. *SIAM Journal on Mathematics of Data Science*, 3(1):171–200, 2021.

Algorithm 1: Geometry-Aligned Weight Selection

Input: Symmetric adjacency matrix $\boldsymbol{W} \in \mathbb{R}^{N\times N}$, geometry time budget T_{geo}.
Output: Mean-one node weights $\boldsymbol{p} \in \mathbb{R}^N_{++}$.

if $W_{ij} \geq 0$ *for all* i, j **then**
  foreach $i \in [N]$ **do**
    $d_i \leftarrow \sum_{j=1}^N W_{ij}$;
  $\boldsymbol{p} \leftarrow N\boldsymbol{d}/(\mathbf{1}^\top \boldsymbol{d})$;
else
  $\boldsymbol{q} \leftarrow \mathbf{0}$, $\boldsymbol{c} \leftarrow \mathbf{0}$, $\mathcal{B} \leftarrow \varnothing$, $t_{\mathrm{best}} \leftarrow +\infty$, $\boldsymbol{q}_{\mathrm{best}} \leftarrow \mathbf{0}$, $\boldsymbol{p} \leftarrow \mathbf{1}$;
  while *the elapsed geometry time is below* T_{geo} **do**
    Compute $t = \phi(\boldsymbol{q})$ and a unit leading eigenvector $\boldsymbol{v}$ of $\boldsymbol{W} - \mathrm{diag}(\boldsymbol{q})$;
    if $t < t_{\mathrm{best}}$ **then**
      $\boldsymbol{q}_{\mathrm{best}} \leftarrow \boldsymbol{q}$, $t_{\mathrm{best}} \leftarrow t$, $\boldsymbol{p} = 1 + \frac{\boldsymbol{q}_{\mathrm{best}}}{t_{\mathrm{best}}}$;
    $\boldsymbol{g} \leftarrow N^{-1}\mathbf{1} - \boldsymbol{v} \odot \boldsymbol{v}$;
    $a \leftarrow t - \boldsymbol{g}^\top \boldsymbol{q}$;
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{(a, \boldsymbol{g})\}$; `// add the spectral cut`
    Update the serious/null-step center $\boldsymbol{c}$ and proximal scale η;
    $\boldsymbol{q} \leftarrow \underset{\mathbf{1}^\top \boldsymbol{z}=0}{\arg\min} \left\{ \max_{(a_s,\boldsymbol{g}_s)\in\mathcal{B}} (a_s + \boldsymbol{g}_s^\top \boldsymbol{z}) + \frac{1}{2\eta}\|\boldsymbol{z} - \boldsymbol{c}\|_2^2 \right\}$;
    Retain at most the prescribed number of cuts in $\mathcal{B}$;
return $\boldsymbol{p}$;

A Time-Limited Spectral-Bundle Solver

For signed graphs, geometry selection reduces to solving problem (5). The function ϕ is convex and may be nondifferentiable when its largest eigenvalue has multiplicity greater than one. The spectral-bundle method represents ϕ by affine cuts generated from leading eigenvectors and uses a proximal model to update the diagonal shift $\boldsymbol{q}$. The complete procedure is summarized in Algorithm 1.

A.1 Spectral cuts

At a query point $\boldsymbol{q}$, let $\boldsymbol{v}$ be a unit leading eigenvector of

$$\boldsymbol{A}(\boldsymbol{q}) = \boldsymbol{W} - \mathrm{diag}(\boldsymbol{q}).$$

For any feasible $\boldsymbol{z}$, the Rayleigh characterization of the largest eigenvalue gives

$$\phi(\boldsymbol{z}) \geq \boldsymbol{v}^\top (\boldsymbol{W} - \mathrm{diag}(\boldsymbol{z}))\, \boldsymbol{v} = \boldsymbol{v}^\top \boldsymbol{W} \boldsymbol{v} - \sum_{i=1}^N v_i^2 z_i. \tag{13}$$

Thus every leading eigenvector supplies a global affine lower bound on ϕ. On the zero-sum subspace $\mathbf{1}^\top \boldsymbol{z} = 0$, its slope can be written as

$$\boldsymbol{g} = \frac{1}{N}\mathbf{1} - \boldsymbol{v} \odot \boldsymbol{v}, \tag{14}$$

because adding a multiple of $\mathbf{1}$ does not change inner products with feasible shifts. The components v_i^2 identify where the current leading spectral direction is concentrated, and the cut uses this information to redistribute the diagonal shift across nodes.

A.2 Proximal bundle model

The method retains a small bundle

$$\mathcal{B} = \{(a_s, \boldsymbol{g}_s)\}_{s=1}^S$$

of spectral cuts. Together they define the piecewise-linear model

$$m_{\mathcal{B}}(\boldsymbol{z}) = \max_{1 \le s \le S} \left(a_s + \boldsymbol{g}_s^\top \boldsymbol{z} \right). \tag{15}$$

Retaining cuts from several query points allows the model to represent multiple leading spectral directions rather than only the direction observed at the current iterate.

Given a bundle center $\boldsymbol{c}$ and proximal scale $\eta > 0$, the next query point solves

$$\boldsymbol{q}^+ = \operatorname*{arg\,min}_{\mathbf{1}^\top \boldsymbol{z} = 0} \left\{ m_{\mathcal{B}}(\boldsymbol{z}) + \frac{1}{2\eta} \|\boldsymbol{z} - \boldsymbol{c}\|_2^2 \right\}. \tag{16}$$

The bundle model drives the spectral objective downward, while the proximal term controls the distance from the center. A serious step moves the center when the new query realizes sufficient objective decrease. A null step keeps the center fixed and adds the new cut, thereby improving the model before the next query. Only the prescribed number of active cuts is retained.

A.3 Restarted Lanczos oracle

Each spectral query requires a leading eigenpair of $\boldsymbol{A}(\boldsymbol{q})$. The implementation approximates this pair with restarted Lanczos iteration (Parlett, 1998). Lanczos builds a Krylov subspace using products of the form

$$\boldsymbol{x} \longmapsto \boldsymbol{W}\boldsymbol{x} - \boldsymbol{q} \odot \boldsymbol{x},$$

so the graph remains sparse throughout the calculation. The Krylov dimension and number of matrix–vector products are bounded for each query. The leading vector from one query warm-starts the next because successive matrices differ only through their diagonal shifts.

Let $(\widehat{\lambda}, \boldsymbol{v})$ be the leading Ritz pair returned by Lanczos. Its residual is

$$r = \left\| \boldsymbol{A}(\boldsymbol{q})\boldsymbol{v} - \widehat{\lambda}\boldsymbol{v} \right\|_2. \tag{17}$$

The residual controls the stopping and restarting of the spectral query. The leading Ritz value estimates $\phi(\boldsymbol{q})$, while the Ritz vector generates the cut in (13).

A.4 Numerical Configuration and Safeguards

Unless otherwise specified, all signed-graph experiments use a geometry budget of $T_{\mathrm{geo}} = 1$ second, including sparse-matrix construction and the spectral-bundle solve. The outer bundle loop is capped at $1{,}000$ iterations, and the deadline is checked between consecutive spectral queries; a bounded Lanczos query that is already in progress is allowed to finish.

Each leading-eigenpair query uses restarted Lanczos with a Krylov dimension of 40, a hard limit of 300 sparse matrix–vector products, and the relative Ritz-residual stopping criterion

$$r \le 10^{-7} \max\{1, |\widehat{\lambda}|\},$$

where r is defined in (17). We use two-pass reorthogonalization and warm-start each query with the leading Ritz vector returned by the preceding query.

The bundle retains at most 12 spectral cuts. The proximal scale in (16) is parameterized as $\eta = Ns$, with initial value $s = 0.1$. A query is accepted as a serious step when its realized objective decrease is at least 0.1 times the model-predicted decrease. After a serious step, s is multiplied by 1.25, up to a maximum of 100; after a null step, it is multiplied by 0.7, down to a minimum of 10^{-4}. The proximal master problem is solved by SLSQP with at most 100 iterations and function tolerance 10^{-11}.

To account for an inexact Ritz pair, we use the residual-corrected shift

$$t_{\mathrm{safe}} = \widehat{\lambda} + \max\{10r, 10^{-7}\}.$$

The corresponding positive loading and normalized geometry are

$$u_i = \max\{q_i + t_{\mathrm{safe}}, 10^{-10}\}, \qquad p_i = \frac{N u_i}{\sum_j u_j}$$

A nonuniform incumbent is retained only when the Lanczos query satisfies its convergence criterion and the corrected objective improves the uniform incumbent by at least 10^{-7}. If no such candidate is accepted before the geometry deadline, the method returns the uniform geometry $\boldsymbol{p} = \mathbf{1}$.

Algorithm 2: Training and inference of recurrent LGA-GNN

Training phase

Input: Training set $\mathcal{D}$, T_{train}, and optimizer OPT.
Output: Trained parameters $\boldsymbol{\Theta}_{\text{train}}$.
for $\boldsymbol{W} \in \mathcal{D}$ **do**
  Select $\boldsymbol{p}$ using Algorithm 1;
  Initialize $\boldsymbol{Y}_0 = \mathrm{Softmax}(\boldsymbol{H}) \in \Delta_k^N$;
  for $t = 1$ **to** T_{train} **do**
    $\boldsymbol{Y}_t \leftarrow \mathcal{F}_{\boldsymbol{\Theta}}(\boldsymbol{Y}_{t-1}; \boldsymbol{W}, \boldsymbol{p})$;
    $\boldsymbol{g} \leftarrow \nabla_{\boldsymbol{\Theta}} f(\boldsymbol{Y}_t; \boldsymbol{W})$;
    $\boldsymbol{\Theta} \leftarrow \mathrm{OPT}(\boldsymbol{\Theta}, \boldsymbol{g})$;
    $\boldsymbol{Y}_t \leftarrow \mathrm{stopgrad}(\boldsymbol{Y}_t)$;
  $\boldsymbol{\Theta}_{\text{train}} \leftarrow \boldsymbol{\Theta}$;

Inference phase

Input: Trained LGA-GNN parameters $\boldsymbol{\Theta}_{\text{train}}$, test graph $\boldsymbol{W}$, T_{infer}.
Output: Best relaxed solution $\boldsymbol{Y}^\star$.
Select $\boldsymbol{p}$ using Algorithm 1;
Initialize $\boldsymbol{Y}_0 = \mathrm{Softmax}(\boldsymbol{H}) \in \Delta_k^N$;
Set $\ell^\star \leftarrow f(\boldsymbol{Y}_0; \boldsymbol{W})$ and $\boldsymbol{Y}^\star \leftarrow \boldsymbol{Y}_0$;
for $t = 1$ **to** T_{infer} **do**
  $\boldsymbol{Y}_t \leftarrow \mathcal{F}_{\boldsymbol{\Theta}_{\text{train}}}(\boldsymbol{Y}_{t-1}; \boldsymbol{W}, \boldsymbol{p})$;
  $\ell_t \leftarrow f(\boldsymbol{Y}_t; \boldsymbol{W})$;
  if $\ell_t < \ell^\star$ **then**
    $(\ell^\star, \boldsymbol{Y}^\star) \leftarrow (\ell_t, \boldsymbol{Y}_t)$;

A.5 BEST-SO-FAR SELECTION

The initial shift is $\boldsymbol{q} = \boldsymbol{0}$, which corresponds exactly to the uniform geometry $\boldsymbol{p} = \boldsymbol{1}$. Its spectral value initializes the incumbent. Every subsequent query replaces the incumbent only when it produces a smaller value of ϕ. Consequently, the best retained value is nonincreasing throughout the run. At the geometry deadline, the selected weights are recovered from the best retained pair by

$$\boldsymbol{p} = \boldsymbol{1} + \frac{\boldsymbol{q}_{\text{best}}}{t_{\text{best}}}, \qquad t_{\text{best}} = \phi(\boldsymbol{q}_{\text{best}}). \tag{18}$$

If no query improves the initial point, then $\boldsymbol{q}_{\text{best}} = \boldsymbol{0}$ and the returned geometry is uniform.

B IMPLEMENTATION DETAILS

Training Datasets. We train `LGA-GNN` on unit-weight two-dimensional random geometric graphs (RGGs). For each graph, the number of nodes is sampled log-uniformly as $N \sim \mathcal{U}_{\log}[10^2, 10^4]$, and the node coordinates are drawn independently and uniformly from $[0, 1]^2$. We sample a target degree d_{tar} uniformly from $\{3, 4, 6, 8, 12, 16, 24, 32\}$ and connect two nodes when their Euclidean distance is at most $\rho = \sqrt{\frac{d_{\text{tar}}}{\pi(N-1)}}$. All edges have unit weight. The training set contains $5{,}000$ batches, each constructed by packing independently generated graphs subject to a node budget of $B = 10{,}000$. Checkpoint selection uses a fixed held-out set of 20 independently generated RGGs with $N \sim \mathcal{U}_{\log}[10^2, 10^3]$ and the same target-degree distribution. These training RGGs are structurally distinct from the testing datasets.

Model Training and Inference. The implementation of the training and inference procedure is shown in Algorithm 2. We train separate LGA-GNN checkpoints for $k = 2$ and $k = 3$. Unless otherwise specified, the shared refinement operator $\mathcal{F}_{\boldsymbol{\Theta}}$ has a hidden dimension of 400 and a depth of 20 layers. The models are trained for 20 epochs with AdamW (Loshchilov & Hutter, 2019) at a learning rate of 10^{-4}, using $T_{\text{train}} = 5$ recurrent refinement steps. Checkpoints are selected using

the fixed held-out RGG validation set described above. All neural computations use single-precision floating point and sparse matrix–matrix aggregation. At inference, the model parameters remain fixed and the shared operator is applied for exactly $T_{\text{infer}} = 100$ steps. We retain the best relaxed solution encountered along the trajectory and recover a discrete solution using the same sampling-based rounding procedure as ROS (Qiu et al., 2025b), drawing 100 samples and selecting the best partition.

Runtime Measurement. Experiments are conducted on an Intel Core i9-12900K CPU and an NVIDIA RTX 3090 GPU. Neural computations and the matrix-multiplication kernels used by the classical baselines are executed on the GPU. CUDA operations are synchronized at all timing boundaries. End-to-end runtime includes graph preprocessing, checkpoint loading when applicable, geometry construction, optimization or recurrent refinement, and discrete rounding. For each instance and k, the computational budget is defined by the measured end-to-end runtime of LGA−GNN with $T_{\text{infer}} = 100$. Baseline comparisons use the best completed solution whose measured end-to-end time does not exceed this budget.

C Proofs for Geometry-Aligned KL

C.1 Proof of Theorem 3.1

Proof. Let $\boldsymbol{D} = \boldsymbol{Y} - \boldsymbol{X}$ and $\boldsymbol{A} = \boldsymbol{P}^{-1/2}\boldsymbol{W}\boldsymbol{P}^{-1/2}$. Since f is quadratic and $\nabla f(\boldsymbol{X}; \boldsymbol{W}) = \boldsymbol{W}\boldsymbol{X}$,

$$f(\boldsymbol{Y}; \boldsymbol{W}) - f(\boldsymbol{X}; \boldsymbol{W}) - \langle \nabla f(\boldsymbol{X}; \boldsymbol{W}), \boldsymbol{D} \rangle = \frac{1}{2} \operatorname{Tr}(\boldsymbol{D}^\top \boldsymbol{W} \boldsymbol{D}).$$

Writing $\boldsymbol{Z} = \boldsymbol{P}^{1/2}\boldsymbol{D}$ and applying the Rayleigh bound to each column gives

$$\frac{1}{2} \operatorname{Tr}(\boldsymbol{D}^\top \boldsymbol{W} \boldsymbol{D}) = \frac{1}{2} \operatorname{Tr}(\boldsymbol{Z}^\top \boldsymbol{A} \boldsymbol{Z}) \le \frac{L_{\boldsymbol{p}}}{2} \|\boldsymbol{Z}\|_F^2.$$

Pinsker's inequality and $\|\boldsymbol{v}\|_1 \ge \|\boldsymbol{v}\|_2$ imply, row by row,

$$D_{\mathrm{KL}}(\boldsymbol{Y}_{i,:} \| \boldsymbol{X}_{i,:}) \ge \frac{1}{2} \|\boldsymbol{Y}_{i,:} - \boldsymbol{X}_{i,:}\|_1^2 \ge \frac{1}{2} \|\boldsymbol{D}_{i,:}\|_2^2.$$

After multiplying by p_i and summing, $B_\omega(\boldsymbol{Y}, \boldsymbol{X}; \boldsymbol{p}) \ge \frac{1}{2} \|\boldsymbol{P}^{1/2}\boldsymbol{D}\|_F^2$. Combining the displays proves (3). Boundary points follow by the standard lower semicontinuous extension of KL divergence. □

C.2 Proof of Theorem 3.2

Proof. We first derive a lower bound that holds for every feasible geometry. Fix $\boldsymbol{p}$, let

$$\boldsymbol{A}_{\boldsymbol{p}} = \boldsymbol{P}^{-1/2}\boldsymbol{W}\boldsymbol{P}^{-1/2}, \qquad \boldsymbol{z} = \boldsymbol{P}^{1/2}\mathbf{1}.$$

The Rayleigh quotient gives

$$L_{\boldsymbol{p}} \ge \frac{\boldsymbol{z}^\top \boldsymbol{A}_{\boldsymbol{p}} \boldsymbol{z}}{\boldsymbol{z}^\top \boldsymbol{z}} = \frac{\mathbf{1}^\top \boldsymbol{W} \mathbf{1}}{\mathbf{1}^\top \boldsymbol{p}} = \frac{\sum_i d_i}{N} = \bar{d}. \tag{19}$$

Thus no feasible geometry can have objective smaller than $\bar{d}$.

Now define $p_i^\star = d_i/\bar{d}$. The standing graph assumptions and nonnegative edge weights give $d_i > 0$, while

$$\mathbf{1}^\top \boldsymbol{p}^\star = \frac{\sum_i d_i}{\bar{d}} = N.$$

Hence $\boldsymbol{p}^\star$ is feasible.

It remains to evaluate its largest eigenvalue. Let $\boldsymbol{P}_\star = \operatorname{diag}(\boldsymbol{p}^\star)$. Since $\boldsymbol{W}\mathbf{1} = \boldsymbol{d} = \bar{d}\,\boldsymbol{p}^\star$,

$$\boldsymbol{P}_\star^{-1/2}\boldsymbol{W}\boldsymbol{P}_\star^{-1/2}\boldsymbol{P}_\star^{1/2}\mathbf{1} = \boldsymbol{P}_\star^{-1/2}\boldsymbol{W}\mathbf{1} = \bar{d}\,\boldsymbol{P}_\star^{1/2}\mathbf{1}. \tag{20}$$

The normalized matrix is entrywise nonnegative, and $\boldsymbol{P}_\star^{1/2}\mathbf{1}$ is strictly positive. Therefore the eigenvalue in (20) is its largest eigenvalue, so

$$L_{\boldsymbol{p}^\star} = \bar{d}.$$

The proposed geometry attains the universal lower bound in (19) and is optimal. □

C.3 Proof of Theorem 3.3

Proof. We prove the equivalence in both directions.

From $\boldsymbol{p}$ to $\boldsymbol{q}$. Fix a feasible $\boldsymbol{p}$, and define

$$t = L_{\boldsymbol{p}}, \qquad \boldsymbol{q} = t(\boldsymbol{p} - \mathbf{1}). \tag{21}$$

The shift has zero sum because

$$\mathbf{1}^\top \boldsymbol{q} = t(\mathbf{1}^\top \boldsymbol{p} - N) = 0.$$

Thus $\boldsymbol{q}$ is feasible for (5).

Let

$$\boldsymbol{A} = \boldsymbol{P}^{-1/2} \boldsymbol{W} \boldsymbol{P}^{-1/2}.$$

Since $t = \lambda_{\max}(\boldsymbol{A})$, the matrix $t\boldsymbol{I} - \boldsymbol{A}$ is positive semidefinite and singular. Congruence by $\boldsymbol{P}^{1/2}$ gives

$$\boldsymbol{P}^{1/2}(t\boldsymbol{I} - \boldsymbol{A})\boldsymbol{P}^{1/2} = t\boldsymbol{P} - \boldsymbol{W} = t\boldsymbol{I} - (\boldsymbol{W} - \operatorname{diag}(\boldsymbol{q})), \tag{22}$$

where the second equality follows from (21). The right-hand side of (22) is also positive semidefinite and singular. Consequently,

$$t = \lambda_{\max}(\boldsymbol{W} - \operatorname{diag}(\boldsymbol{q})) = \phi(\boldsymbol{q}).$$

The map from $\boldsymbol{p}$ to $\boldsymbol{q}$ therefore preserves both feasibility and the objective value.

From $\boldsymbol{q}$ to $\boldsymbol{p}$. Fix a feasible $\boldsymbol{q}$, and define

$$t = \phi(\boldsymbol{q}), \qquad \boldsymbol{p} = \mathbf{1} + \frac{\boldsymbol{q}}{t}. \tag{23}$$

First, $t > 0$. Indeed,

$$\operatorname{tr}(\boldsymbol{W} - \operatorname{diag}(\boldsymbol{q})) = -\mathbf{1}^\top \boldsymbol{q} = 0.$$

The shifted matrix is nonzero under the standing graph assumptions. A nonzero symmetric matrix with zero trace has a positive eigenvalue, so its largest eigenvalue t is positive.

Second, the recovered weights have the required normalization:

$$\mathbf{1}^\top \boldsymbol{p} = N + \frac{\mathbf{1}^\top \boldsymbol{q}}{t} = N.$$

Third, the recovered weights are strictly positive. By the definition of t,

$$t\boldsymbol{I} - (\boldsymbol{W} - \operatorname{diag}(\boldsymbol{q})) = t\boldsymbol{P} - \boldsymbol{W} \tag{24}$$

is positive semidefinite. Its i-th diagonal entry is tp_i, so $p_i \geq 0$. If $p_i = 0$, a positive-semidefinite matrix with a zero i-th diagonal entry has a zero i-th row and column. The off-diagonal entries in (24) would then imply $W_{ij} = 0$ for every j, contradicting the standing graph assumptions. Hence $p_i > 0$ for every i, and $\boldsymbol{p}$ is feasible for (4).

Finally, the matrix in (24) is singular because t is an eigenvalue of $\boldsymbol{W} - \operatorname{diag}(\boldsymbol{q})$. Applying the inverse congruence gives the positive-semidefinite singular matrix

$$t\boldsymbol{I} - \boldsymbol{P}^{-1/2} \boldsymbol{W} \boldsymbol{P}^{-1/2}.$$

Therefore

$$L_{\boldsymbol{p}} = \lambda_{\max}\left(\boldsymbol{P}^{-1/2} \boldsymbol{W} \boldsymbol{P}^{-1/2}\right) = t = \phi(\boldsymbol{q}).$$

The reverse map also preserves feasibility and the objective value, which proves the equivalence and the recovery formula. □

C.4 Proof of Corollary 3.4

Proof. For a nonnegative graph, Theorem 3.2 returns a global minimizer of (4). Since $\boldsymbol{p} = \mathbf{1}$ is feasible, its objective cannot exceed $L_{\mathbf{1}}$.

Now consider the signed-graph branch. The uniform geometry $\boldsymbol{p} = \mathbf{1}$ corresponds under Theorem 3.3 to $\boldsymbol{q} = \mathbf{0}$, and

$$\phi(\mathbf{0}) = \lambda_{\max}(\boldsymbol{W}) = L_{\mathbf{1}}.$$

Algorithm 1 initializes the best-so-far iterate at $\boldsymbol{q} = \mathbf{0}$ and replaces it only when a smaller value of ϕ is found. Therefore

$$\phi(\boldsymbol{q}_{\text{best}}) \le \phi(\mathbf{0}).$$

The recovery result in Theorem 3.3 gives

$$L_{\boldsymbol{p}} = \phi(\boldsymbol{q}_{\text{best}}) \le \phi(\mathbf{0}) = L_{\mathbf{1}}.$$

If no improving iterate is found, then $\boldsymbol{q}_{\text{best}} = \mathbf{0}$ and equality holds. If the incumbent is replaced by an iterate with a smaller objective, the inequality is strict. □

C.5 Proof of Proposition 3.5

Proof. The mirror subproblem separates across rows. For row i, introduce a multiplier ν_i for $\mathbf{1}^\top \boldsymbol{y}_i = 1$. Ignoring terms independent of $\boldsymbol{y}_i$, the optimality condition is

$$(\boldsymbol{W}\boldsymbol{X}^{(m)})_{ic} + \frac{p_i}{\eta}\left(\log Y_{ic} - \log X_{ic}^{(m)} + 1\right) + \nu_i = 0.$$

Exponentiation gives $Y_{ic} \propto X_{ic}^{(m)} \exp[-\eta p_i^{-1}(\boldsymbol{W}\boldsymbol{X}^{(m)})_{ic}]$. Normalizing each row gives exactly the softmax operation in (7). □

C.6 Proof of Theorem 3.6

Proof. For each $m \ge 0$, write

$$\boldsymbol{X}^{(m+1)} = \mathcal{T}(\boldsymbol{X}^{(m)}; \eta_m, \boldsymbol{p}), \qquad \eta_m = (C_W + 1)^{-1}.$$

Because $\boldsymbol{X}^{(m)}$ is feasible for the mirror subproblem defining $\boldsymbol{X}^{(m+1)}$, comparison with this feasible point gives

$$\left\langle \nabla f(\boldsymbol{X}^{(m)}; \boldsymbol{W}), \boldsymbol{X}^{(m+1)} - \boldsymbol{X}^{(m)} \right\rangle + \eta_m^{-1} B_\omega\left(\boldsymbol{X}^{(m+1)}, \boldsymbol{X}^{(m)}; \boldsymbol{p}\right) \le 0. \tag{25}$$

By Corollary 3.4, the selected geometry satisfies

$$L_{\boldsymbol{p}} \le \lambda_{\max}(\boldsymbol{W}) \le \|\boldsymbol{W}\|_2 \le \|\boldsymbol{W}\|_{1,1} \le C_W. \tag{26}$$

In particular,

$$\eta_m^{-1} - L_{\boldsymbol{p}} = C_W + 1 - L_{\boldsymbol{p}} \ge 1 > 0.$$

Applying Theorem 3.1 with $\boldsymbol{X} = \boldsymbol{X}^{(m)}$ and $\boldsymbol{Y} = \boldsymbol{X}^{(m+1)}$ yields

$$f(\boldsymbol{X}^{(m+1)}; \boldsymbol{W}) \le f(\boldsymbol{X}^{(m)}; \boldsymbol{W}) + \left\langle \nabla f(\boldsymbol{X}^{(m)}; \boldsymbol{W}), \boldsymbol{X}^{(m+1)} - \boldsymbol{X}^{(m)} \right\rangle + L_{\boldsymbol{p}} B_\omega\left(\boldsymbol{X}^{(m+1)}, \boldsymbol{X}^{(m)}; \boldsymbol{p}\right).$$

Combining this inequality with (25) gives

$$f(\boldsymbol{X}^{(m+1)}; \boldsymbol{W}) \le f(\boldsymbol{X}^{(m)}; \boldsymbol{W}) - (C_W + 1 - L_{\boldsymbol{p}})\, B_\omega\left(\boldsymbol{X}^{(m+1)}, \boldsymbol{X}^{(m)}; \boldsymbol{p}\right). \tag{27}$$

Summing (27) over $m = 0, \dots, M-1$ and telescoping the objective values gives

$$(C_W + 1 - L_{\boldsymbol{p}}) \sum_{m=0}^{M-1} B_\omega\left(\boldsymbol{X}^{(m+1)}, \boldsymbol{X}^{(m)}; \boldsymbol{p}\right) \le f(\boldsymbol{X}^{(0)}; \boldsymbol{W}) - f(\boldsymbol{X}^{(M)}; \boldsymbol{W}). \tag{28}$$

Since $\boldsymbol{X}^{(M)} \in \Delta_k^N$,

$$f(\boldsymbol{X}^{(M)}; \boldsymbol{W}) \geq \inf_{\boldsymbol{X} \in \Delta_k^N} f(\boldsymbol{X}; \boldsymbol{W}).$$

Hence,

$$\sum_{m=0}^{M-1} B_\omega\Big(\boldsymbol{X}^{(m+1)}, \boldsymbol{X}^{(m)}; \boldsymbol{p}\Big) \leq \frac{f(\boldsymbol{X}^{(0)}; \boldsymbol{W}) - \inf_{\boldsymbol{X} \in \Delta_k^N} f(\boldsymbol{X}; \boldsymbol{W})}{C_W + 1 - L_{\boldsymbol{p}}}. \tag{29}$$

It remains to bound the objective range. For every $\boldsymbol{X} \in \Delta_k^N$,

$$0 \leq \langle \boldsymbol{X}_{i,:}, \boldsymbol{X}_{j,:} \rangle \leq 1.$$

Therefore,

$$|f(\boldsymbol{X}; \boldsymbol{W})| \leq \frac{1}{2} \sum_{i,j=1}^{N} |W_{ij}| = \frac{1}{2} \|\boldsymbol{W}\|_{1,1} \leq \frac{C_W}{2}.$$

It follows that

$$f(\boldsymbol{X}^{(0)}; \boldsymbol{W}) - \inf_{\boldsymbol{X} \in \Delta_k^N} f(\boldsymbol{X}; \boldsymbol{W}) \leq C_W. \tag{30}$$

Combining (29) and (30), and using the nonnegativity of the Bregman divergence, gives

$$\min_{0 \leq m < M} B_\omega\Big(\boldsymbol{X}^{(m+1)}, \boldsymbol{X}^{(m)}; \boldsymbol{p}\Big) \leq \frac{1}{M} \sum_{m=0}^{M-1} B_\omega\Big(\boldsymbol{X}^{(m+1)}, \boldsymbol{X}^{(m)}; \boldsymbol{p}\Big) \leq \frac{C_W}{M(C_W + 1 - L_{\boldsymbol{p}})}.$$

Finally, substituting

$$\boldsymbol{X}^{(m+1)} = \mathcal{T}(\boldsymbol{X}^{(m)}; \eta_m, \boldsymbol{p})$$

proves the stated result. □

D Representation Proofs

D.1 Proof of Theorem 4.1

Proof. We first state three auxiliary lemmas. Throughout the proof, $\boldsymbol{p}$ is the geometry selected for the given graph and $\boldsymbol{P} = \mathrm{diag}(\boldsymbol{p})$. Since $\boldsymbol{p} \in \mathbb{R}^N_{++}$, the matrix $\boldsymbol{P}^{-1}\boldsymbol{W}$ is a fixed linear operator. Thus, the selected weighted-KL geometry changes only the linear term in the usual entropy mirror-descent map, and the original representation argument applies with $\boldsymbol{W}$ replaced by $\boldsymbol{P}^{-1}\boldsymbol{W}$.

Lemma D.1 (Local Lipschitz continuity of $\mathcal{T}$)**.** *For any $\eta > 0$, any $\boldsymbol{p} \in \mathbb{R}^N_{++}$, and any strictly positive $\boldsymbol{X}^\star \in \mathbb{R}^{N \times k}_{>0}$, there exist a neighborhood $\mathcal{U} \subset \mathbb{R}^{N \times k}_{>0}$ of $\boldsymbol{X}^\star$ and a constant $L > 0$ such that, for all $\boldsymbol{Z}, \boldsymbol{Z}' \in \mathcal{U}$,*

$$\|\mathcal{T}(\boldsymbol{Z}; \eta, \boldsymbol{p}) - \mathcal{T}(\boldsymbol{Z}'; \eta, \boldsymbol{p})\|_F \leq L \|\boldsymbol{Z} - \boldsymbol{Z}'\|_F.$$

Proof. On $\mathbb{R}^{N \times k}_{>0}$, the map $\boldsymbol{Z} \mapsto \log \boldsymbol{Z}$ is smooth, the map $\boldsymbol{Z} \mapsto -\eta \boldsymbol{P}^{-1}\boldsymbol{W}\boldsymbol{Z}$ is linear, and row-wise softmax is smooth. Hence $\mathcal{T}(\cdot; \eta, \boldsymbol{p})$ is smooth on positive matrices and is locally Lipschitz on a compact neighborhood of $\boldsymbol{X}^\star$. □

Lemma D.2 (Softmax channel suppression)**.** *Let $\boldsymbol{\alpha} \in \mathbb{R}^k$ and $\boldsymbol{\beta} \in \mathbb{R}^q$ satisfy $\max_j \beta_j \leq -\gamma < 0$. Define*

$$\mathrm{Softmax}([\boldsymbol{\alpha} \quad T\boldsymbol{\beta}]) = [\boldsymbol{c}(T) \quad \boldsymbol{d}(T)] \in \mathbb{R}^{k+q}.$$

Then, with $Z_0 = \sum_{i=1}^k e^{\alpha_i}$,

$$\|\boldsymbol{d}(T)\|_2 \leq \frac{q e^{-\gamma T}}{Z_0}, \qquad \|\boldsymbol{c}(T) - \mathrm{Softmax}(\boldsymbol{\alpha})\|_2 \leq \frac{q e^{-\gamma T}}{Z_0}.$$

In particular, both quantities can be made arbitrarily small by taking T sufficiently large.

Proof. Let $Z(T) = Z_0 + \sum_{j=1}^{q} e^{T\beta_j}$. Since $\beta_j \le -\gamma$,

$$\|\boldsymbol{d}(T)\|_2 \le \|\boldsymbol{d}(T)\|_1 = \frac{\sum_{j=1}^{q} e^{T\beta_j}}{Z(T)} \le \frac{qe^{-\gamma T}}{Z_0}. \tag{31}$$

Moreover,

$$c_i(T) = \frac{e^{\alpha_i}}{Z(T)} = \frac{Z_0}{Z(T)}\,\mathrm{Softmax}(\boldsymbol{\alpha})_i.$$

Therefore,

$$\|\boldsymbol{c}(T) - \mathrm{Softmax}(\boldsymbol{\alpha})\|_2 \le \frac{\sum_{j=1}^{q} e^{T\beta_j}}{Z(T)} \le \frac{qe^{-\gamma T}}{Z_0},$$

where the last inequality follows as in (31). □

Lemma D.3 (Positive preservation)**.** *If $\boldsymbol{X}^{(0)} \in \Delta_k^N$ is strictly positive, then every iterate in the GA-KL mirror trajectory $\{\boldsymbol{X}^{(m)}\}_{m=0}^{M}$ is strictly positive.*

Proof. The logarithm and the linear term in (7) are finite at a strictly positive iterate, and softmax has strictly positive outputs. The claim follows by induction. □

We now proceed by induction on the layer index m.

Base case ($m = 0$). Set $\boldsymbol{Y}^{(0)} = \boldsymbol{X}^{(0)}$. Since $d_0 = k$, the claim holds with zero error.

Induction step. Assume that for some $m \in \{0, \dots, M-1\}$ and some $\varepsilon_{\mathrm{align},m} > 0$, the parameters of the first m layers have been chosen so that

$$\left\| \begin{bmatrix} \boldsymbol{X}^{(m)} & \boldsymbol{0}_{N\times(d_m-k)} \end{bmatrix} - \boldsymbol{Y}^{(m)} \right\|_F^2 \le \varepsilon_{\mathrm{align},m}.$$

Write

$$\boldsymbol{Y}^{(m)} = \begin{bmatrix} \boldsymbol{R}^{(m)} & \boldsymbol{S}^{(m)} \end{bmatrix},$$

where $\boldsymbol{R}^{(m)} \in \mathbb{R}^{N\times k}$ and $\boldsymbol{S}^{(m)} \in \mathbb{R}^{N\times(d_m-k)}$. Then

$$\|\boldsymbol{R}^{(m)} - \boldsymbol{X}^{(m)}\|_F^2 \le \varepsilon_{\mathrm{align},m}, \qquad \|\boldsymbol{S}^{(m)}\|_F^2 \le \varepsilon_{\mathrm{align},m}. \tag{32}$$

We construct $\lambda_m = \eta_m$, $\boldsymbol{U}^{(m)}$ and $\boldsymbol{V}^{(m)}$ as

$$\boldsymbol{U}^{(m)} = \begin{pmatrix} \boldsymbol{I}_k & T\boldsymbol{1}_{k\times(d_{m+1}-k)} \\ \boldsymbol{0}_{(d_m-k)\times k} & \boldsymbol{0}_{(d_m-k)\times(d_{m+1}-k)} \end{pmatrix}, \quad \boldsymbol{V}^{(m)} = \begin{pmatrix} \boldsymbol{I}_k & \boldsymbol{0}_{k\times(d_{m+1}-k)} \\ \boldsymbol{0}_{(d_m-k)\times k} & \boldsymbol{0}_{(d_m-k)\times(d_{m+1}-k)} \end{pmatrix} \tag{33}$$

where $T > 0$ will be chosen below. Substitution into (8) gives the pre-softmax logits

$$\begin{bmatrix} \log \boldsymbol{R}^{(m)} - \eta_m \boldsymbol{P}^{-1}\boldsymbol{W}\boldsymbol{R}^{(m)} & T \log \boldsymbol{R}^{(m)} \boldsymbol{1}_{k\times(d_{m+1}-k)} \end{bmatrix}. \tag{34}$$

The first k channels are exactly the selected-geometry weighted-KL mirror-descent logits applied to $\boldsymbol{R}^{(m)}$; this is the only place where the selected geometry enters the construction.

Denote the post-softmax output by $\boldsymbol{Y}^{(m+1)} = [\boldsymbol{R}^{(m+1)}\ \boldsymbol{S}^{(m+1)}]$. By Lemmas D.1 and D.3, there is a constant $L_m > 0$ such that, whenever $\boldsymbol{R}^{(m)}$ is sufficiently close to $\boldsymbol{X}^{(m)}$,

$$\left\| \mathcal{T}(\boldsymbol{X}^{(m)}; \eta_m, \boldsymbol{p}) - \mathcal{T}(\boldsymbol{R}^{(m)}; \eta_m, \boldsymbol{p}) \right\|_F \le L_m \|\boldsymbol{X}^{(m)} - \boldsymbol{R}^{(m)}\|_F.$$

For row i, let

$$\boldsymbol{\alpha}_i = \left[\log \boldsymbol{R}^{(m)} - \eta_m \boldsymbol{P}^{-1}\boldsymbol{W}\boldsymbol{R}^{(m)} \right]_{i,:}, \qquad \beta_i = \sum_{c=1}^{k} \log R_{ic}^{(m)}.$$

Every row of $\boldsymbol{R}^{(m)}$ is strictly positive and has total mass at most one. For $k \ge 2$, this implies $0 < R_{ic}^{(m)} < 1$ and hence $\beta_i < 0$. Because there are finitely many rows, $\gamma_m := -\max_i \beta_i > 0$.

Applying Lemma D.2 row-wise with the repeated vector $\beta_i \mathbf{1}_{d_{m+1}-k}$, we may choose T so that, for any prescribed $\tau > 0$,

$$\left\| \boldsymbol{R}^{(m+1)} - \mathcal{T}(\boldsymbol{R}^{(m)}; \eta_m, \boldsymbol{p}) \right\|_F \le \sqrt{N}\,\tau, \qquad \|\boldsymbol{S}^{(m+1)}\|_F \le \sqrt{N}\,\tau.$$

When $d_{m+1} = k$, these bounds hold with $\tau = 0$. Consequently,

$$\|\boldsymbol{X}^{(m+1)} - \boldsymbol{R}^{(m+1)}\|_F \le L_m \sqrt{\varepsilon_{\mathrm{align},m}} + \sqrt{N}\,\tau.$$

For any target tolerance $\varepsilon_{\mathrm{align},m+1} > 0$, choose

$$\varepsilon_{\mathrm{align},m} \le \frac{(2-\sqrt{3})\varepsilon_{\mathrm{align},m+1}}{2L_m^2}, \qquad \tau \le \frac{\sqrt{\varepsilon_{\mathrm{align},m+1}}}{2\sqrt{N}}.$$

Then

$$\|\boldsymbol{X}^{(m+1)} - \boldsymbol{R}^{(m+1)}\|_F^2 \le \frac{3\varepsilon_{\mathrm{align},m+1}}{4}, \qquad \|\boldsymbol{S}^{(m+1)}\|_F^2 \le \frac{\varepsilon_{\mathrm{align},m+1}}{4},$$

and therefore

$$\left\| \begin{bmatrix} \boldsymbol{X}^{(m+1)} & \mathbf{0} \end{bmatrix} - \boldsymbol{Y}^{(m+1)} \right\|_F^2 \le \varepsilon_{\mathrm{align},m+1}.$$

Starting with $\varepsilon_{\mathrm{align},M} = \varepsilon_{\mathrm{align}}$ and choosing the preceding tolerances backward sufficiently small makes every layer error at most $\varepsilon_{\mathrm{align}}$. This completes the induction and proves (9). □

D.2 Proof of Theorem 4.2

Proof. By Theorem 3.6, the analytical GA-KL iterates $\{\boldsymbol{X}^{(m)}\}_{m=0}^{M}$ with $\eta = (C_W + 1)^{-1}$ satisfy

$$\min_{0 \le m < M} B_\omega\Big(\mathcal{T}(\boldsymbol{X}^{(m)}; \eta, \boldsymbol{p}), \boldsymbol{X}^{(m)}; \boldsymbol{p}\Big) \le \frac{C_W}{M(C_W + 1 - L_{\boldsymbol{p}})}. \tag{35}$$

Because $\boldsymbol{p}$ is returned by Algorithm 1, Corollary 3.4 gives

$$L_{\boldsymbol{p}} \le L_{\mathbf{1}} = \lambda_{\max}(\boldsymbol{W}) \le \|\boldsymbol{W}\|_F \le \|\boldsymbol{W}\|_{1,1} \le C_W.$$

Hence $C_W + 1 - L_{\boldsymbol{p}} \ge 1$. Choosing

$$M = \left\lceil \frac{2C_W}{\varepsilon_{\mathrm{stat}}} \right\rceil$$

therefore ensures that, for some $m_\star \in \{0, \dots, M-1\}$,

$$B_\omega\Big(\mathcal{T}(\boldsymbol{X}^{(m_\star)}; \eta, \boldsymbol{p}), \boldsymbol{X}^{(m_\star)}; \boldsymbol{p}\Big) \le \frac{\varepsilon_{\mathrm{stat}}}{2}. \tag{36}$$

We next control the effect of the normalized readout. By Theorem 4.1, for any prescribed $\varepsilon_{\mathrm{align}} > 0$, the aligned construction can be chosen so that, for every m,

$$\left\| \boldsymbol{Y}^{(m)}_{:,1:k} - \boldsymbol{X}^{(m)} \right\|_F^2 \le \varepsilon_{\mathrm{align}}. \tag{37}$$

Let the total softmax mass assigned to the extra channels in row i be

$$r_i^{(m)} := \sum_{j=k+1}^{d_m} Y_{ij}^{(m)}.$$

By the channel-suppression construction in the proof of Theorem 4.1, for any $\tau > 0$ the finite suppression factors can be chosen so that $r_i^{(m)} \le \tau$ for every i and m. Since

$$\widehat{\boldsymbol{Y}}^{(m)}_{i,:} - \boldsymbol{Y}^{(m)}_{i,1:k} = r_i^{(m)} \widehat{\boldsymbol{Y}}^{(m)}_{i,:},$$

and $\|\widehat{\boldsymbol{Y}}^{(m)}_{i,:}\|_2 \le 1$, we have

$$\left\| \widehat{\boldsymbol{Y}}^{(m)} - \boldsymbol{Y}^{(m)}_{:,1:k} \right\|_F^2 \le \sum_{i=1}^{N} \big(r_i^{(m)}\big)^2 \le N\tau^2. \tag{38}$$

Combining (37) and (38) gives

$$\left\|\widehat{\boldsymbol{Y}}^{(m)} - \boldsymbol{X}^{(m)}\right\|_F \le \sqrt{\varepsilon_{\text{align}}} + \sqrt{N}\,\tau. \tag{39}$$

Define the GA-KL residual function

$$\rho(\boldsymbol{Z}) \coloneqq B_\omega(\mathcal{T}(\boldsymbol{Z};\eta,\boldsymbol{p}),\boldsymbol{Z};\boldsymbol{p})\,.$$

By Lemma D.3, the finite analytical trajectory is strictly positive. On a compact neighborhood of this trajectory in the relative interior of Δ_k^N, the maps $\log$, $\mathcal{T}$, and B_ω are smooth. Thus there is a finite constant $K > 0$ such that, for every m and every $\boldsymbol{Z}$ in this neighborhood,

$$\left|\rho(\boldsymbol{Z}) - \rho(\boldsymbol{X}^{(m)})\right| \le K\left\|\boldsymbol{Z} - \boldsymbol{X}^{(m)}\right\|_F. \tag{40}$$

Choose $\varepsilon_{\text{align}}$ and τ sufficiently small that all normalized outputs lie in this neighborhood and

$$K\left(\sqrt{\varepsilon_{\text{align}}} + \sqrt{N}\,\tau\right) \le \frac{\varepsilon_{\text{stat}}}{2}.$$

Applying (39) and (40) at $m = m_\star$, and then using (36), yields

$$B_\omega\Big(\mathcal{T}(\widehat{\boldsymbol{Y}}^{(m_\star)};\eta,\boldsymbol{p}),\widehat{\boldsymbol{Y}}^{(m_\star)};\boldsymbol{p}\Big) \le \varepsilon_{\text{stat}}.$$

The alignment construction preserves the depth M and the prescribed layer widths. Choosing $d_0 = d_M = k$ and $k \le d_m \le d_{\max}$ for $0 < m < M$ therefore gives the claimed bounded-width realization.

□

D.3 Justification of Remark 4.3

Proof. In the constructive proof of Theorem 4.1, the identity blocks in (33) and the choice $\lambda_m = \eta$ depend only on the layer index m within the aligned block. They do not depend on the recurrent step. The selected geometry $\boldsymbol{p} = \boldsymbol{p}(\boldsymbol{W})$ is supplied as a graph-dependent input through $\boldsymbol{P}^{-1}\boldsymbol{W}$ and does not alter these trainable parameter templates.

Unrolling the shared M-layer LGA-GNN block for T_{train} recurrent steps yields a depth-MT_{train} network. For every fixed layer position $m \in \{0,\dots,M-1\}$, that position occurs only finitely many times over

$$(\boldsymbol{W}, t) \in \mathcal{D} \times \{0,\dots,T_{\text{train}}-1\}.$$

Because the training set and recurrent horizon are finite and all analytical GA-KL states are strictly positive, the finitely many local Lipschitz constants and channel-suppression requirements in the proof of Theorem 4.1 admit common finite bounds. Therefore, for each layer position m, one can choose a single suppression factor and hence a single parameter tuple in (33) that works for every recurrent occurrence and every graph in $\mathcal{D}$.

Applying the same inductive alignment argument to the resulting unrolled shared-block network shows that, for any $\varepsilon > 0$, a shared recurrent parametrization can reproduce all analytical GA-KL states entering the finite objective (11) within ε. The function $f(\cdot;\boldsymbol{W})$ is continuous on the compact simplex, and the objective contains only finitely many terms. Its value under the shared recurrent LGA-GNN can therefore be made arbitrarily close to $\mathcal{L}_{\text{GAKL}}$. In particular, for every $\delta > 0$, there exists $\boldsymbol{\Theta}$ such that

$$\mathcal{L}_{\boldsymbol{\Theta}} \le \mathcal{L}_{\text{GAKL}} + \delta.$$

Taking the infimum over $\boldsymbol{\Theta}$ and then letting $\delta \downarrow 0$ proves (12). □

E Ablation Study

E.1 Effect of Geometry Alignment on Signed Graphs

We ablate the proposed geometry alignment on the 36 signed Gset instances. For each value of k, both variants use the same pretrained checkpoint, initialization seed, 100 recurrent refinement

Table 1: Geometry ablation on the 36 signed Gset instances. Gain denotes the mean paired reduction in normalized objective, with a bootstrap 95% confidence interval over instances. The paired p-value is computed using a two-sided Wilcoxon signed-rank test. Times are averaged over the 36 instances.

	Normalized objective (%) ↓		Paired comparison		End-to-end time (s) ↓	
k	Uniform	Spectral-Bundle	Gain (pp) [95% CI]	p-value	Uniform	Spectral-Bundle
2	-24.05	$\mathbf{-24.19}$	0.141 [0.066, 0.220]	0.0020	3.210	4.179
3	-28.34	$\mathbf{-28.52}$	0.182 [0.094, 0.277]	0.0015	3.268	4.261

Table 2: Comparison between LGA−GNN and the matched pretrained GNN on real-world graphs. Values are mean normalized objectives (%) ± one sample standard deviation over 10 independent runs. The better mean for each setting is shown in bold. The Mean row first averages the six instances within each run and then aggregates across runs.

	$k = 2$		$k = 3$	
Instance	GNN	LGA−GNN	GNN	LGA−GNN
Cora	19.511 ± 0.099	$\mathbf{18.885 \pm 0.242}$	$\mathbf{3.358 \pm 0.084}$	3.484 ± 0.060
CiteSeer	15.798 ± 0.143	$\mathbf{15.306 \pm 0.148}$	2.474 ± 0.053	$\mathbf{2.427 \pm 0.068}$
Email-Eu-core	38.763 ± 0.039	$\mathbf{38.290 \pm 0.020}$	26.437 ± 3.136	$\mathbf{20.964 \pm 0.022}$
CA-GrQc	44.180 ± 0.458	$\mathbf{33.991 \pm 0.091}$	31.434 ± 0.967	$\mathbf{17.731 \pm 0.211}$
p2p-Gnutella08	20.539 ± 0.064	$\mathbf{20.397 \pm 0.206}$	$\mathbf{5.787 \pm 0.067}$	5.796 ± 0.069
Wiki-Vote	28.432 ± 0.177	$\mathbf{27.800 \pm 0.015}$	33.003 ± 8.877	$\mathbf{15.056 \pm 0.053}$
Mean	27.870 ± 0.077	$\mathbf{25.778 \pm 0.048}$	17.082 ± 1.781	$\mathbf{10.910 \pm 0.046}$

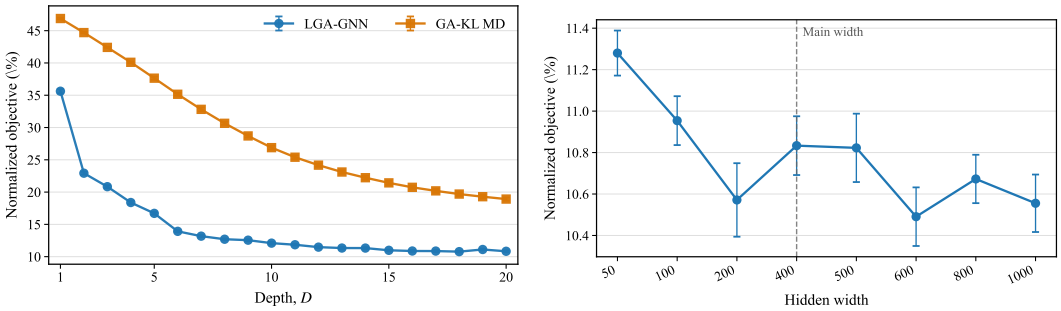
steps, and 100 rounding samples. The UNIFORM variant sets $p_i = 1$, whereas SPECTRAL-BUNDLE computes the node weights by solving the proposed geometry-selection problem with a one-second budget. Geometry construction is included in the end-to-end time. We report the instance-wise mean normalized Min-k-Partition objective.

As shown in Table 1, the proposed geometry consistently improves the average normalized objective for both partition settings. The mean paired reductions are 0.141 and 0.182 percentage points for $k = 2$ and $k = 3$, respectively, while geometry selection adds approximately one second of preprocessing. We additionally apply a two-sided paired Wilcoxon signed-rank test to the per-instance normalized-objective differences. The improvements are statistically significant for both $k = 2$ ($p = 0.0020$) and $k = 3$ ($p = 0.0015$). These results isolate the benefit of geometry alignment from the learned checkpoint and the test-time refinement procedure.

E.2 COMPARISON WITH A MATCHED PRETRAINED GNN

To isolate the contribution of the lifted geometry-aligned update, we compare LGA−GNN with a matched pretrained GNN on the six real-world graphs. The two models use the same training and validation datasets, hidden dimension, depth, training schedule, recurrent training and inference horizons, row-wise softmax output parameterization, optimizer, checkpoint-selection rule, initialization, and rounding procedure. The only architectural difference lies in the intermediate update block: the matched model replaces the lifted GA-KL update with a generic message-passing GNN and therefore contains no analytical mirror map. Both methods use $T_{\text{infer}} = 100$, 100 rounding samples, and the same ten inference seeds.

As shown in Table 2, LGA−GNN improves upon the matched GNN (Morris et al., 2019) in 10 of the 12 instance–k settings, including all six graphs for $k = 2$. Averaged across instances, it reduces the normalized objective by 2.09 percentage points for $k = 2$ and 6.17 for $k = 3$. The largest gains occur on CA-GrQc and Wiki-Vote, whereas the two advantages of GNN for $k = 3$ are only 0.13 and 0.01 percentage points on Cora and p2p-Gnutella08, respectively.



(a) Depth ablation.

(b) Width ablation.

Figure 3: Architecture ablations on a fixed 3-regular RRG with $N = 10{,}000$ and $k = 2$. Markers and error bars report the mean and one standard deviation over 10 runs. (a) LGA-GNN is compared with exactly D analytical GA-KL MD iterations. (b) The hidden width is varied with $D = 20$, and the dashed line marks the width used by the main model.

Table 3: Training-unroll ablation on a 3-regular RRG with $N = 10{,}000$ and $k = 2$. Values are means $\pm$ one sample standard deviation over 10 runs.

T_{train}	Epochs	Normalized objective (%) $\downarrow$	End-to-end time (s)
1	100	8.8300 ± 0.0478	5.261 ± 0.015
2	50	8.2340 ± 0.0454	5.263 ± 0.012
4	25	8.3307 ± 0.0820	5.265 ± 0.004
5	20	8.2640 ± 0.0626	5.267 ± 0.009

E.3 Effect of Learned Update Depth

We conduct the architecture and training ablations on a fixed 3-regular RRG with $N = 10{,}000$ and $k = 2$. To isolate network depth, we fix the hidden width to 400, set $T_{\text{train}} = T_{\text{infer}} = 1$, and vary depth $D \in \{1, \ldots, 20\}$. We compare each learned update with exactly D iterations of the analytical GA-KL mirror-descent update using aligned initialization and rounding seeds.

Figure 3a shows a clear improvement with learned update depth: the objective decreases from 35.62 at $D = 1$ to 10.83 at $D = 20$.

The gain is rapid in early layers and saturates around $D = 15$–20. LGA-GNN outperforms GA-KL MD at every tested depth, and a four-layer learned update already surpasses 20 analytical iterations. At $D = 20$, LGA-GNN improves upon its analytical counterpart by 8.09 percentage points. These results demonstrate that latent lifting produces a substantially more expressive update than simply stacking additional analytical iterations.

E.4 Effect of Hidden Width

We fix $D = 20$ and $T_{\text{train}} = T_{\text{infer}} = 1$, and vary the hidden width over $\{50, 100, 200, 400, 500, 600, 800, 1000\}$. All variants use the same training data, optimizer, training schedule, and checkpoint-selection rule.

As shown in Figure 3b, increasing width from 50 to 200 reduces the normalized objective from 11.28 to 10.57. Performance then enters a narrow plateau, with the best mean objective of 10.49 obtained at width 600 and no consistent gain from wider models. In contrast, the end-to-end time increases from 1.06 seconds at width 50 to 1.36 seconds at width 1000. The validation-selected width of 400 therefore provides a compact intermediate capacity, while the wider variants offer only modest and non-monotonic quality changes at higher cost.

Table 4: Test-time refinement ablation on a 3-regular RRG with $N = 10{,}000$ and $k = 2$. Values are means $\pm$ one sample standard deviation over 10 runs.

T_{infer}	Normalized objective (%) $\downarrow$	End-to-end time (s) $\downarrow$
1	10.9260 ± 0.1315	1.121 ± 0.009
5	9.1013 ± 0.0667	1.306 ± 0.016
10	8.8207 ± 0.0820	1.525 ± 0.053
20	8.5747 ± 0.0718	2.011 ± 0.281
50	8.3493 ± 0.0865	3.172 ± 0.016
100	8.2753 ± 0.0492	5.262 ± 0.013

Table 5: Training-set-size ablation on a 3-regular RRG with $N = 10{,}000$ and $k = 2$. Objectives are reported as means $\pm$ one sample standard deviation over 10 runs.

Training fraction	Epochs	Normalized objective (%) $\downarrow$
50%	40	8.4567 ± 0.0607
75%	27	8.4460 ± 0.0679
100%	20	8.2400 ± 0.0599

E.5 Effect of Training Unroll

We vary $T_{\text{train}} \in \{1, 2, 4, 5\}$ while fixing $T_{\text{infer}} = 100$. To approximately match the number of recurrent applications during training, the corresponding models are trained for 100, 50, 25, and 20 epochs, respectively. All other settings are held fixed.

Table 3 shows that increasing T_{train} from 1 to 2 improves the objective by 0.60 percentage points. Performance then remains stable, with $T_{\text{train}} = 2$ and 5 differing by only 0.03 percentage points. Because all variants share the same 100-step inference architecture, their test times are effectively identical. The result confirms that exposing the model to a refinement trajectory during training is more important than increasing the training unroll indefinitely.

E.6 Effect of Test-Time Refinement

We fix $T_{\text{train}} = 5$ and evaluate $T_{\text{infer}} \in \{1, 5, 10, 20, 50, 100\}$ without early stopping. This directly measures the quality–runtime trade-off enabled by the shared recurrent operator.

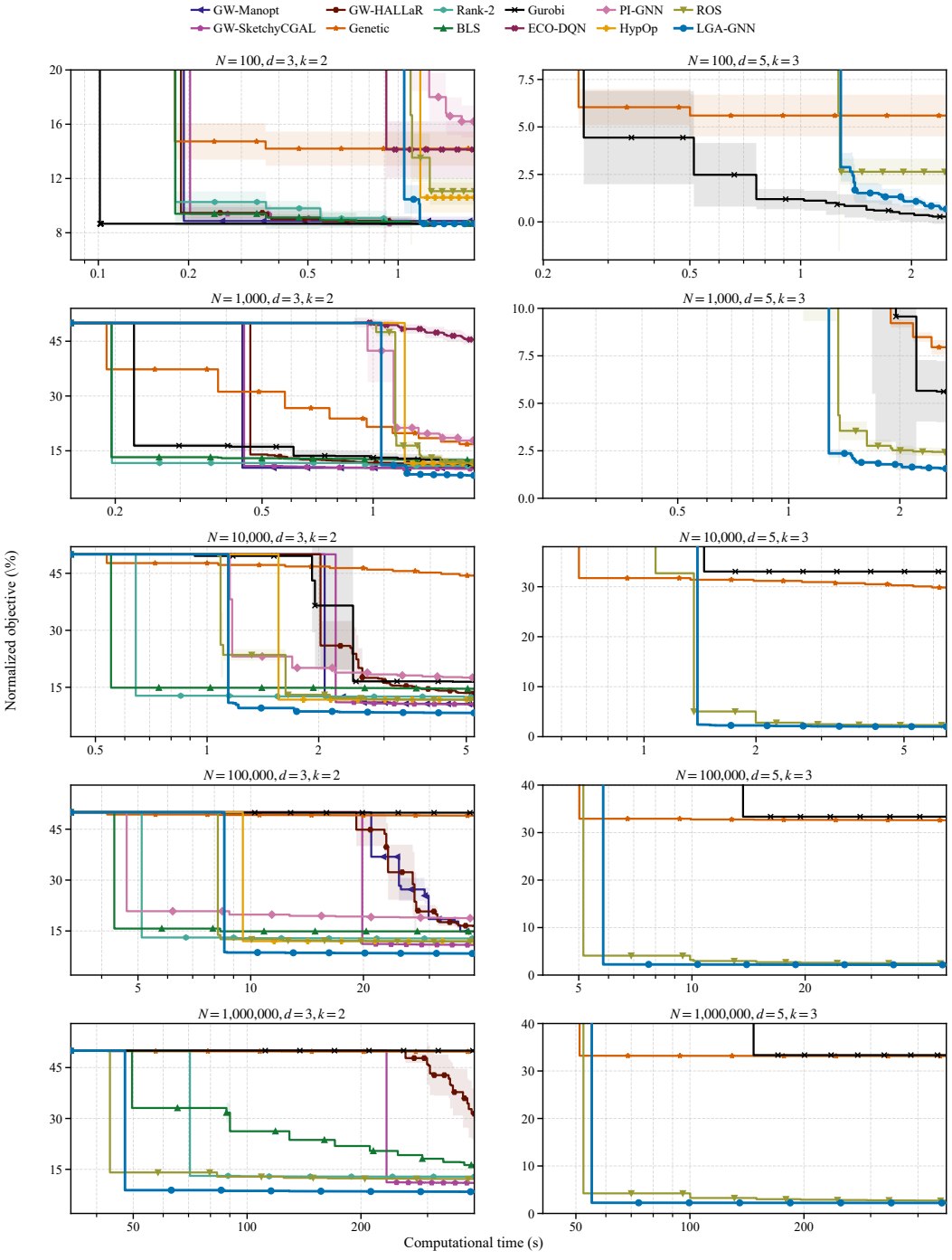
Table 4 shows that every increase in inference depth improves solution quality. The first five steps recover 69% of the total 100-step improvement, while 50 steps recover 97%. Continuing to $T_{\text{infer}} = 100$ yields the best final objective, reducing it by 2.65 percentage points relative to one-step inference. LGA-GNN therefore provides a smooth quality–runtime trade-off and can exploit longer computational budget without retraining.

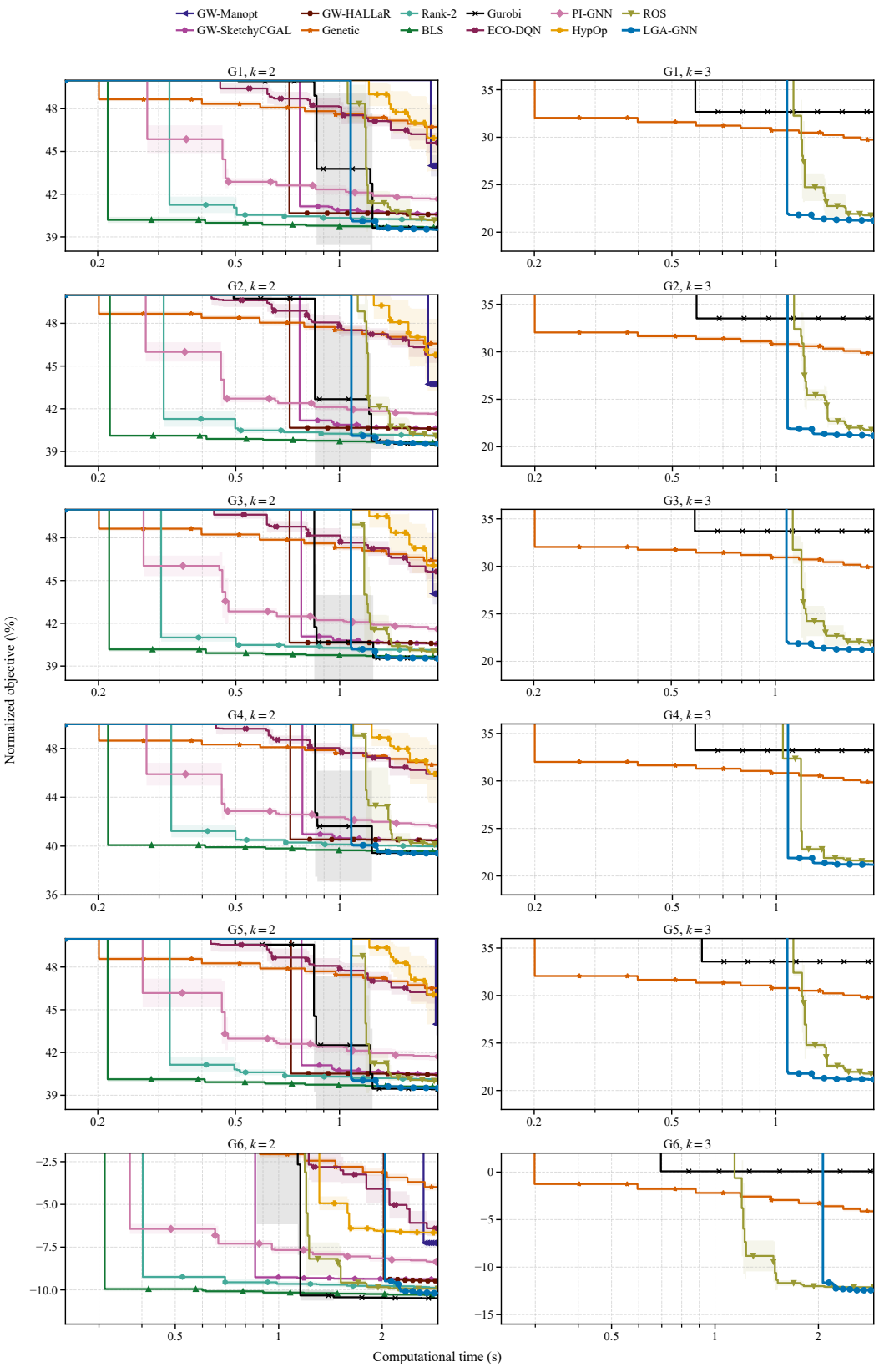
E.7 Effect of Training-Set Size

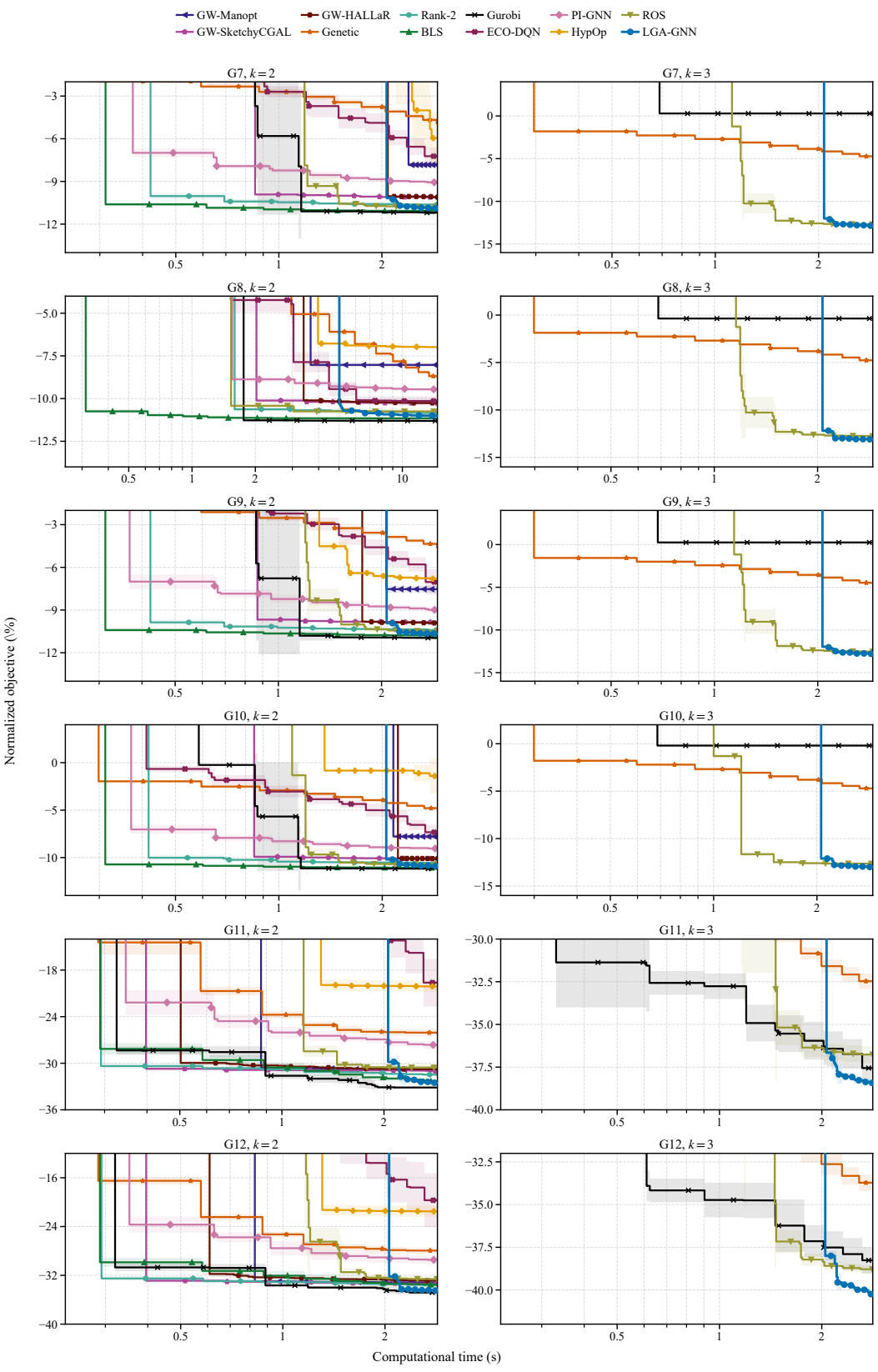
We train with nested subsets containing 50%, 75%, and 100% of the training graphs while fixing $D = 20$, width $= 400$, $T_{\text{train}} = 5$, and $T_{\text{infer}} = 100$. The three variants use 40, 27, and 20 epochs, respectively, approximately matching the total number of graph examples observed during training.

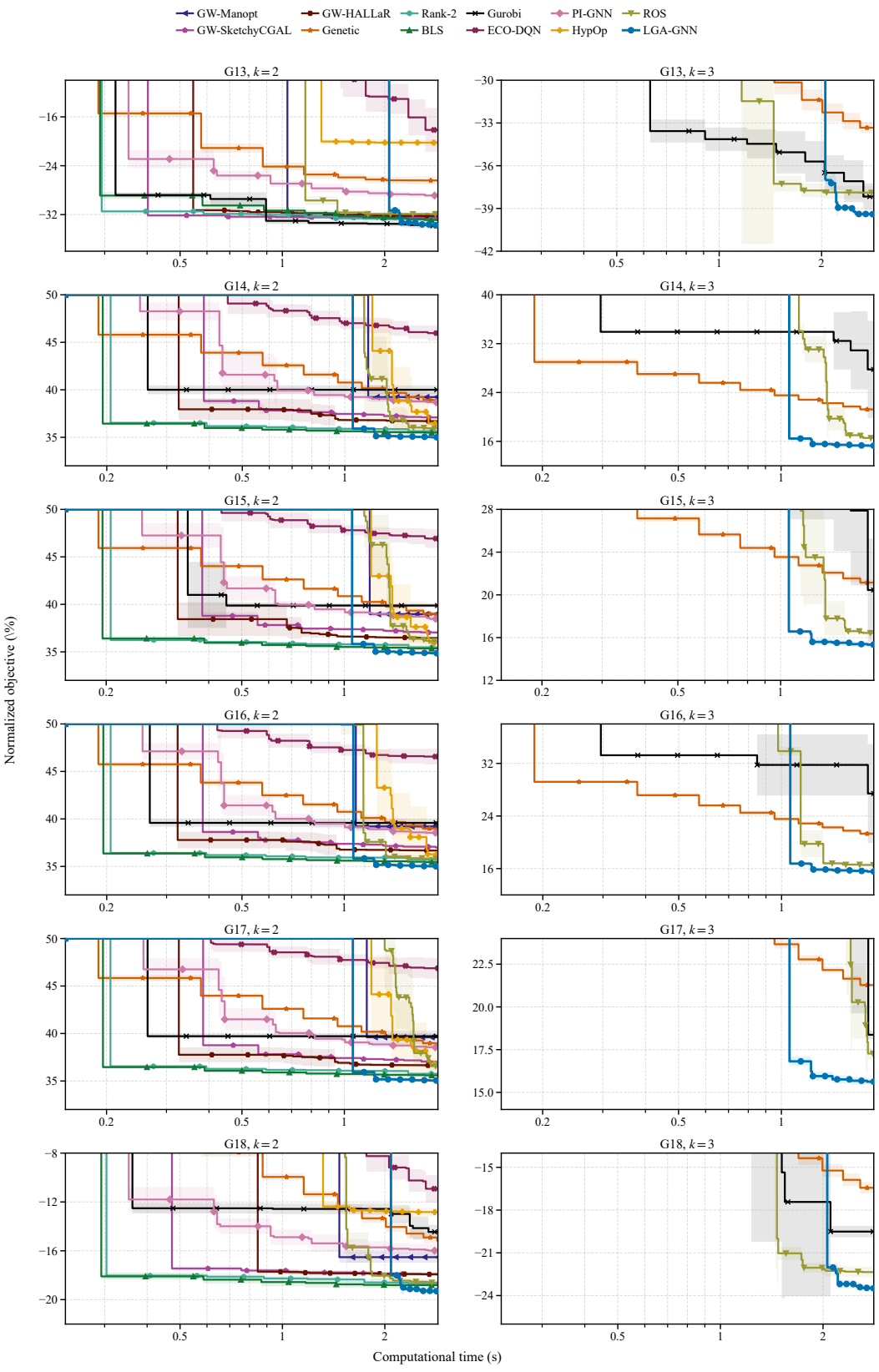
As shown in Table 5, full-data training achieves the best objective, improving upon the 50% and 75% subsets by 0.22 and 0.21 percentage points, respectively. Because the epoch schedule controls the total number of observed graph examples, this improvement isolates the benefit of greater training-graph diversity rather than additional training computation.

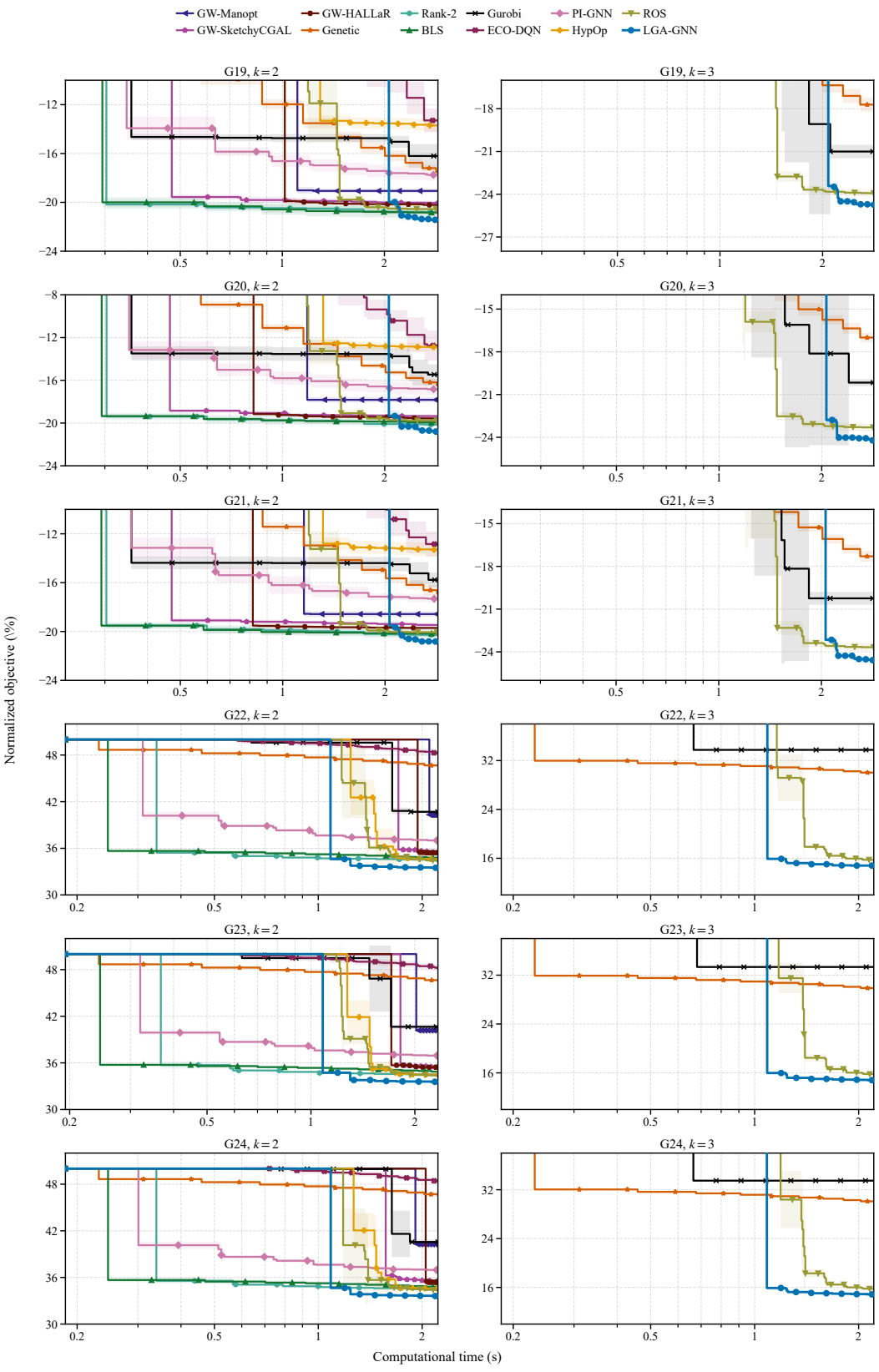
F Detailed Results on Each Instance

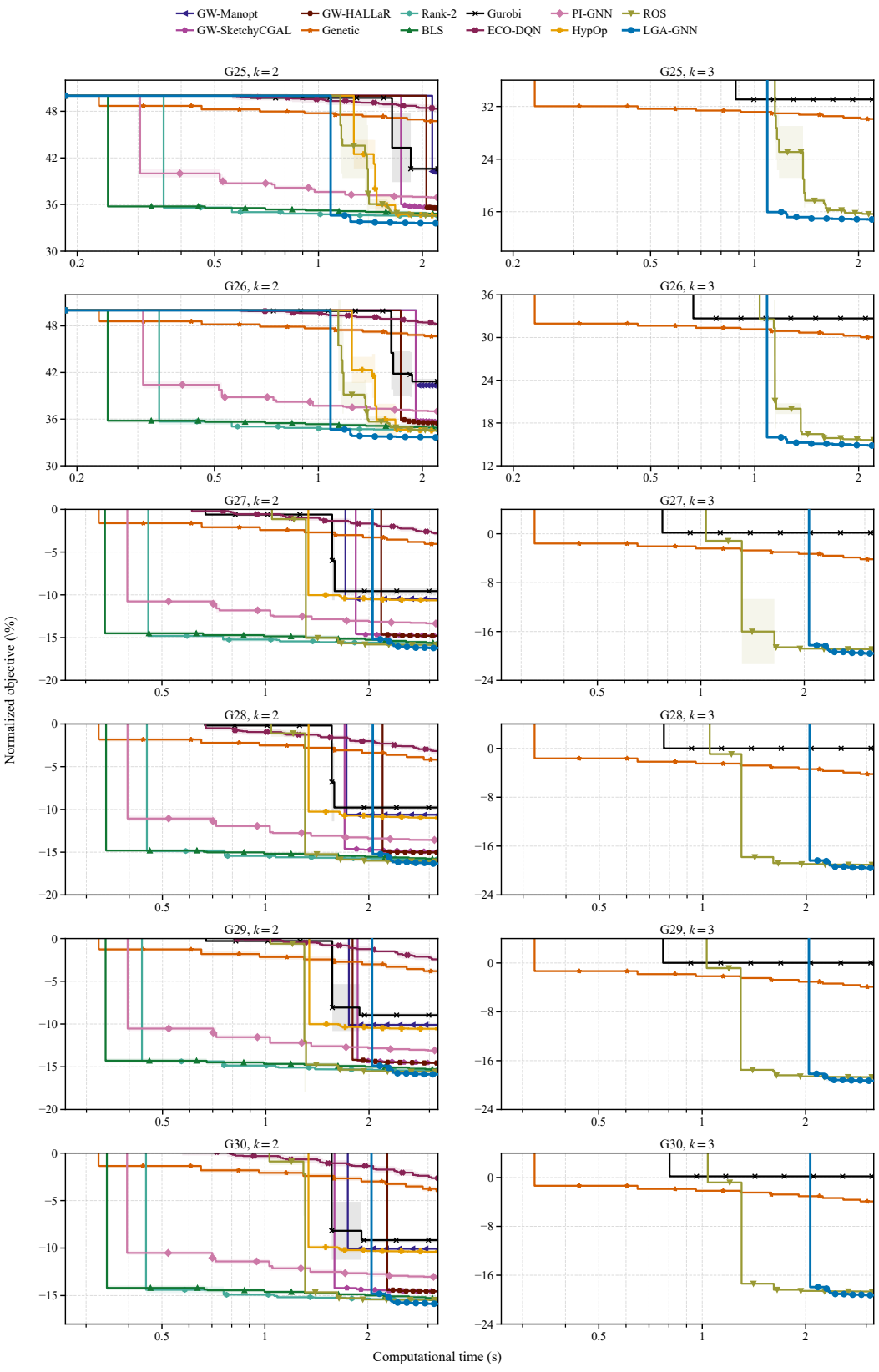


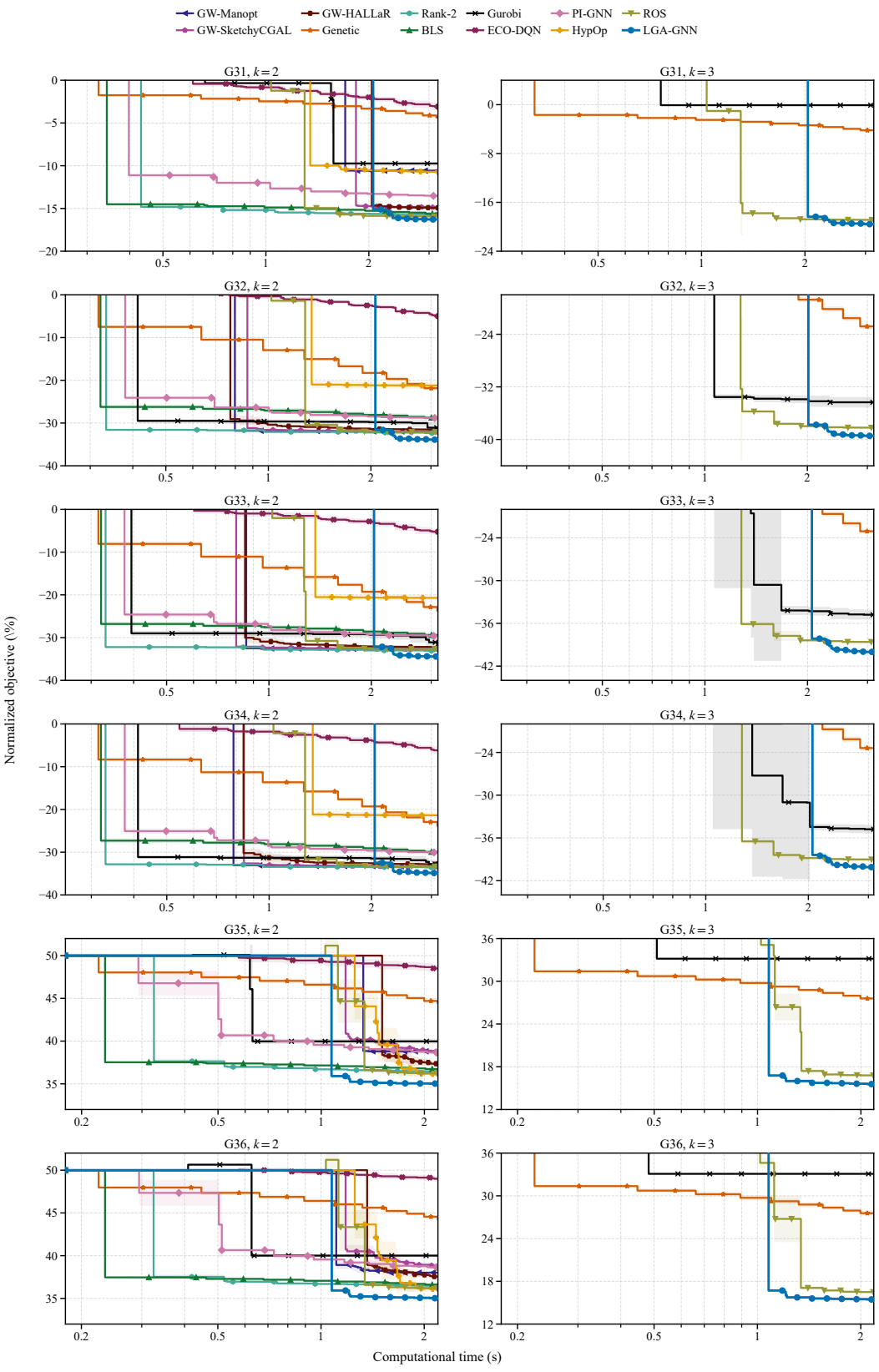

GW-Manopt
GW-SketchyCGAL
GW-HALLaR
Genetic
Rank-2
BLS
Gurobi
ECO-DQN
PI-GNN
HypOp
ROS
LGA-GNN
G1, k = 2
G1, k = 3
G2, k = 2
G2, k = 3
G3, k = 2
G3, k = 3
G4, k = 2
G4, k = 3
G5, k = 2
G5, k = 3
G6, k = 2
G6, k = 3
Normalized objective (%)
Computational time (s)

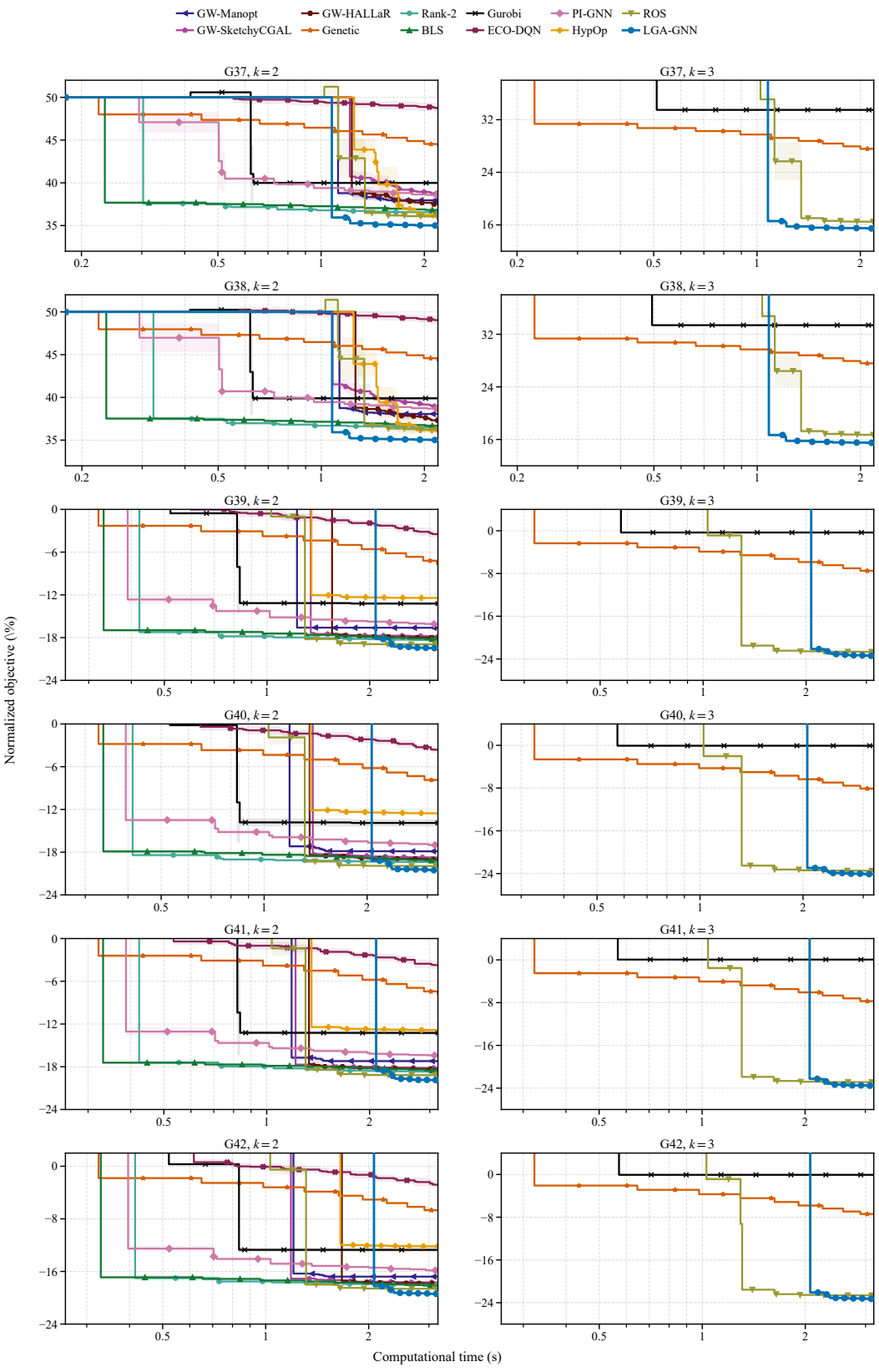

GW-Manopt
GW-SketchyCGAL
GW-HALLaR
Genetic
Rank-2
BLS
Gurobi
ECO-DQN
PI-GNN
HypOp
ROS
LGA-GNN
G7, k = 2
G7, k = 3
G8, k = 2
G8, k = 3
G9, k = 2
G9, k = 3
G10, k = 2
G10, k = 3
G11, k = 2
G11, k = 3
G12, k = 2
G12, k = 3
Normalized objective (%)
Computational time (s)

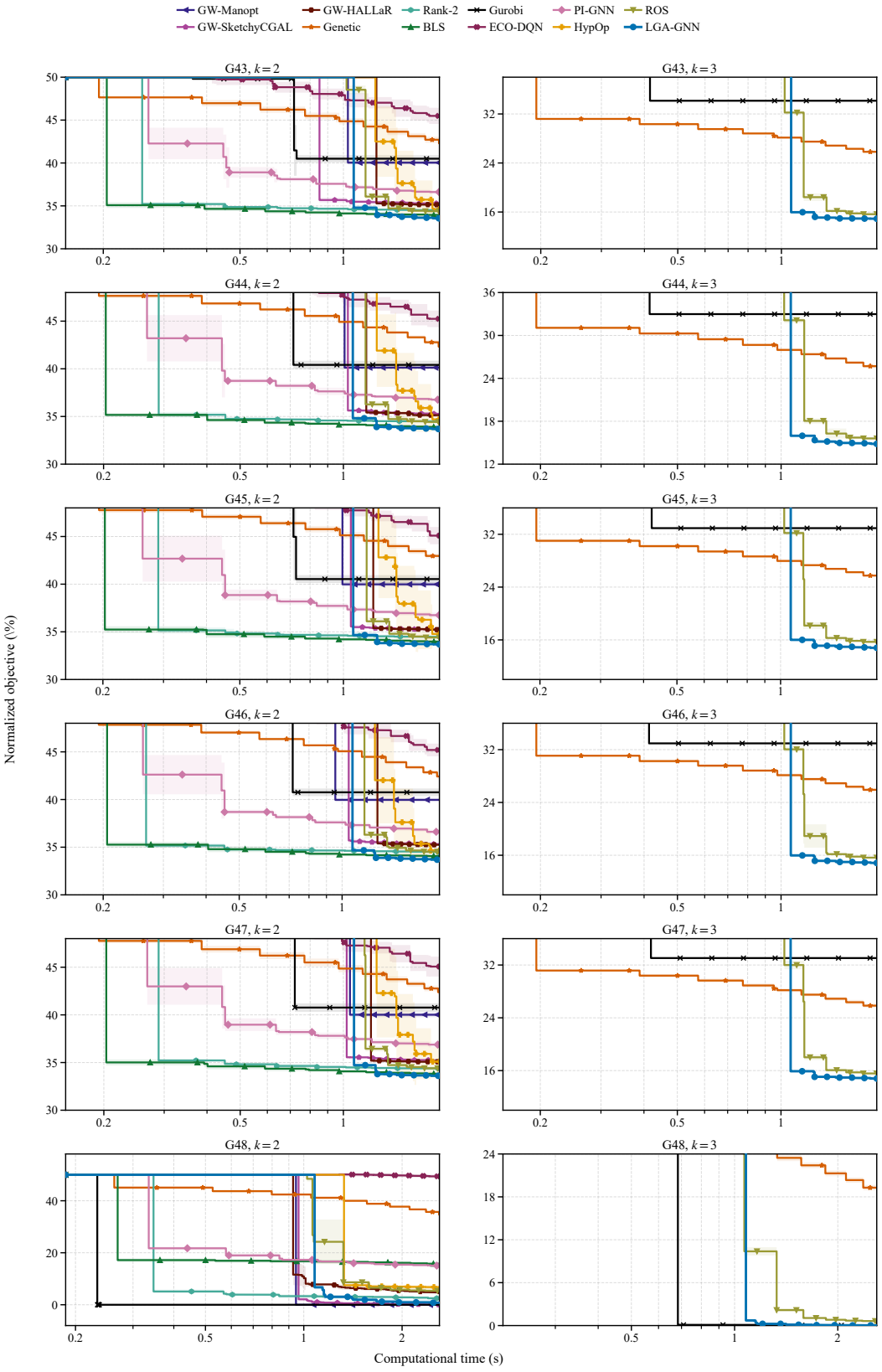

GW-Manopt
GW-SketchyCGAL
GW-HALLaR
Genetic
Rank-2
BLS
Gurobi
ECO-DQN
PI-GNN
HypOp
ROS
LGA-GNN
G13, k = 2
G13, k = 3
G14, k = 2
G14, k = 3
G15, k = 2
G15, k = 3
G16, k = 2
G16, k = 3
G17, k = 2
G17, k = 3
G18, k = 2
G18, k = 3
Normalized objective (%)
Computational time (s)

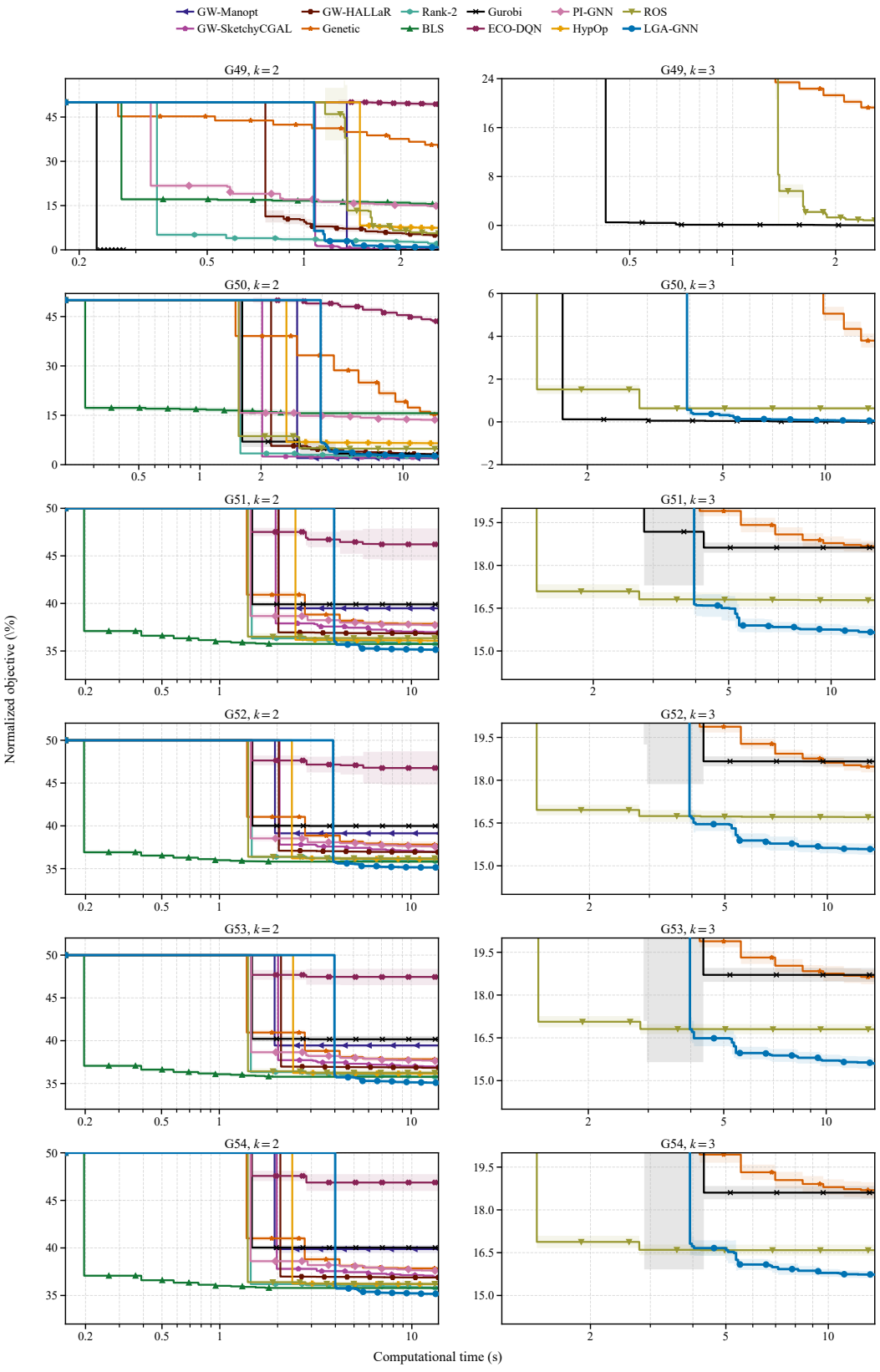

GW-Manopt
GW-SketchyCGAL
GW-HALLaR
Genetic
Rank-2
BLS
Gurobi
ECO-DQN
PI-GNN
HypOp
ROS
LGA-GNN
G19, k = 2
G19, k = 3
G20, k = 2
G20, k = 3
G21, k = 2
G21, k = 3
G22, k = 2
G22, k = 3
G23, k = 2
G23, k = 3
G24, k = 2
G24, k = 3
Normalized objective (%)
Computational time (s)

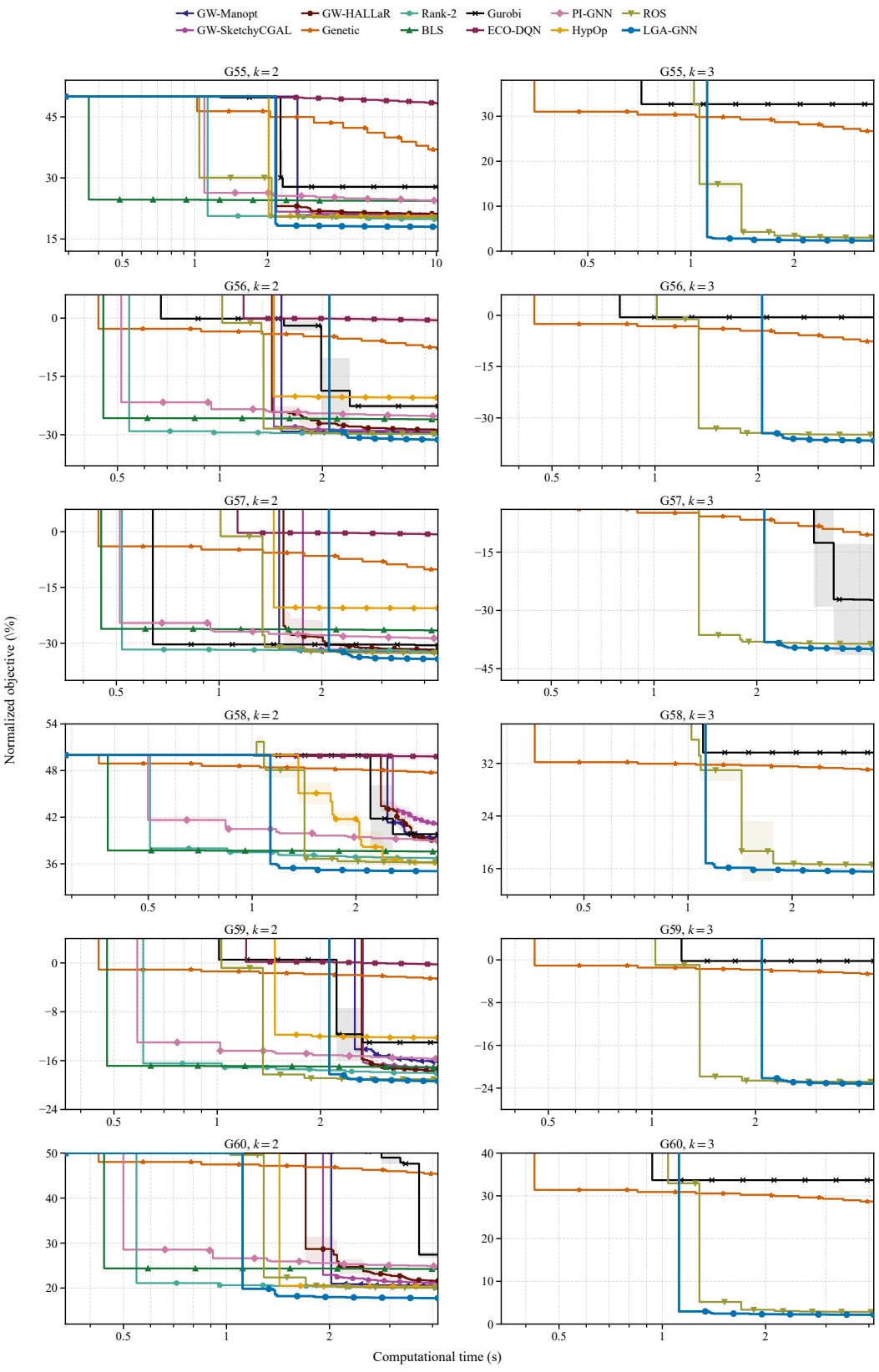

GW-Manopt
GW-SketchyCGAL
GW-HALLaR
Genetic
Rank-2
BLS
Gurobi
ECO-DQN
PI-GNN
HypOp
ROS
LGA-GNN
G25, k = 2
G25, k = 3
G26, k = 2
G26, k = 3
G27, k = 2
G27, k = 3
G28, k = 2
G28, k = 3
G29, k = 2
G29, k = 3
G30, k = 2
G30, k = 3
Normalized objective (%)
Computational time (s)

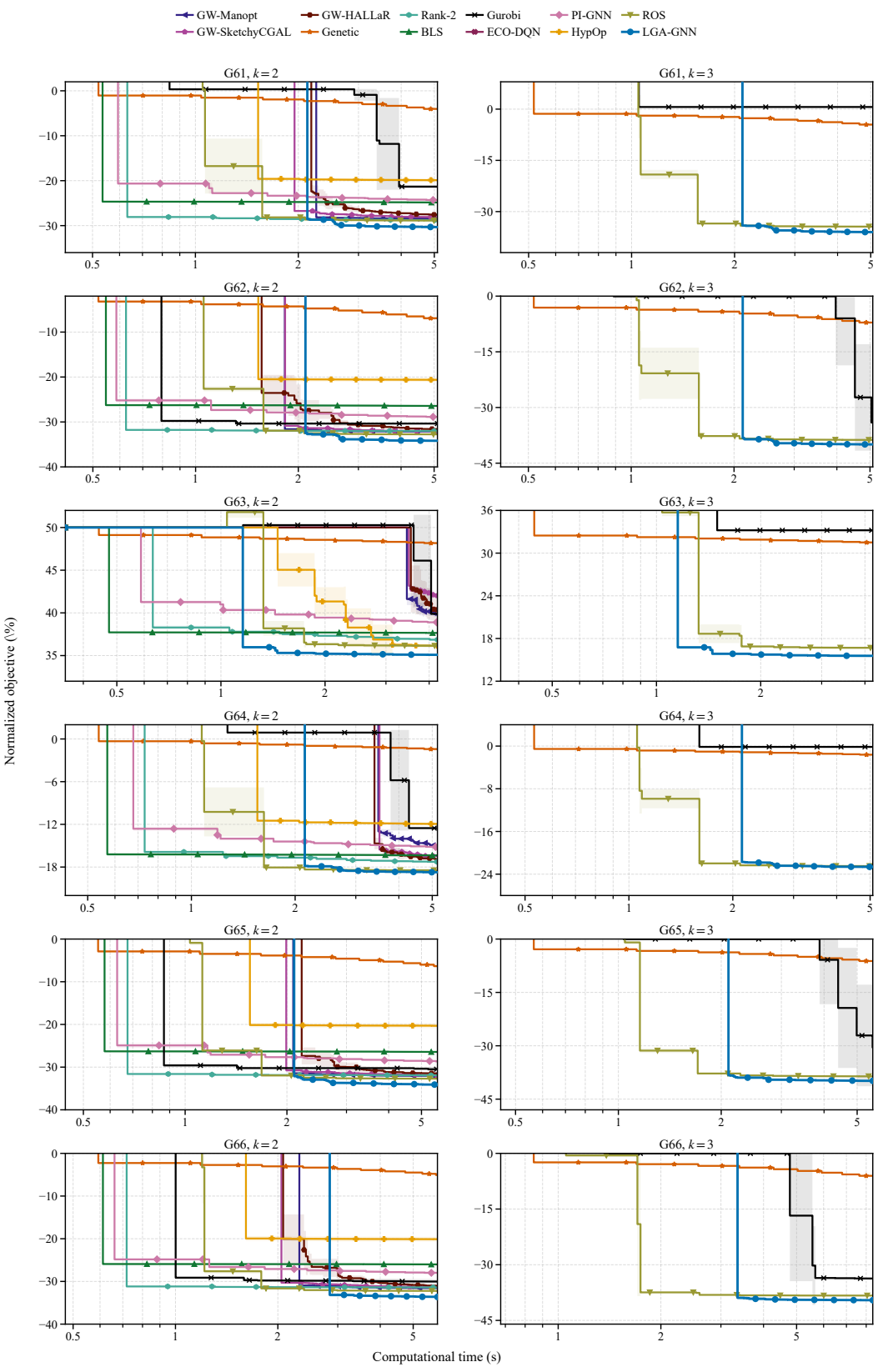

GW-Manopt
GW-SketchyCGAL
GW-HALLaR
Genetic
Rank-2
BLS
Gurobi
ECO-DQN
PI-GNN
HypOp
ROS
LGA-GNN
G31, k = 2
G31, k = 3
G32, k = 2
G32, k = 3
G33, k = 2
G33, k = 3
G34, k = 2
G34, k = 3
G35, k = 2
G35, k = 3
G36, k = 2
G36, k = 3
Normalized objective (%)
Computational time (s)

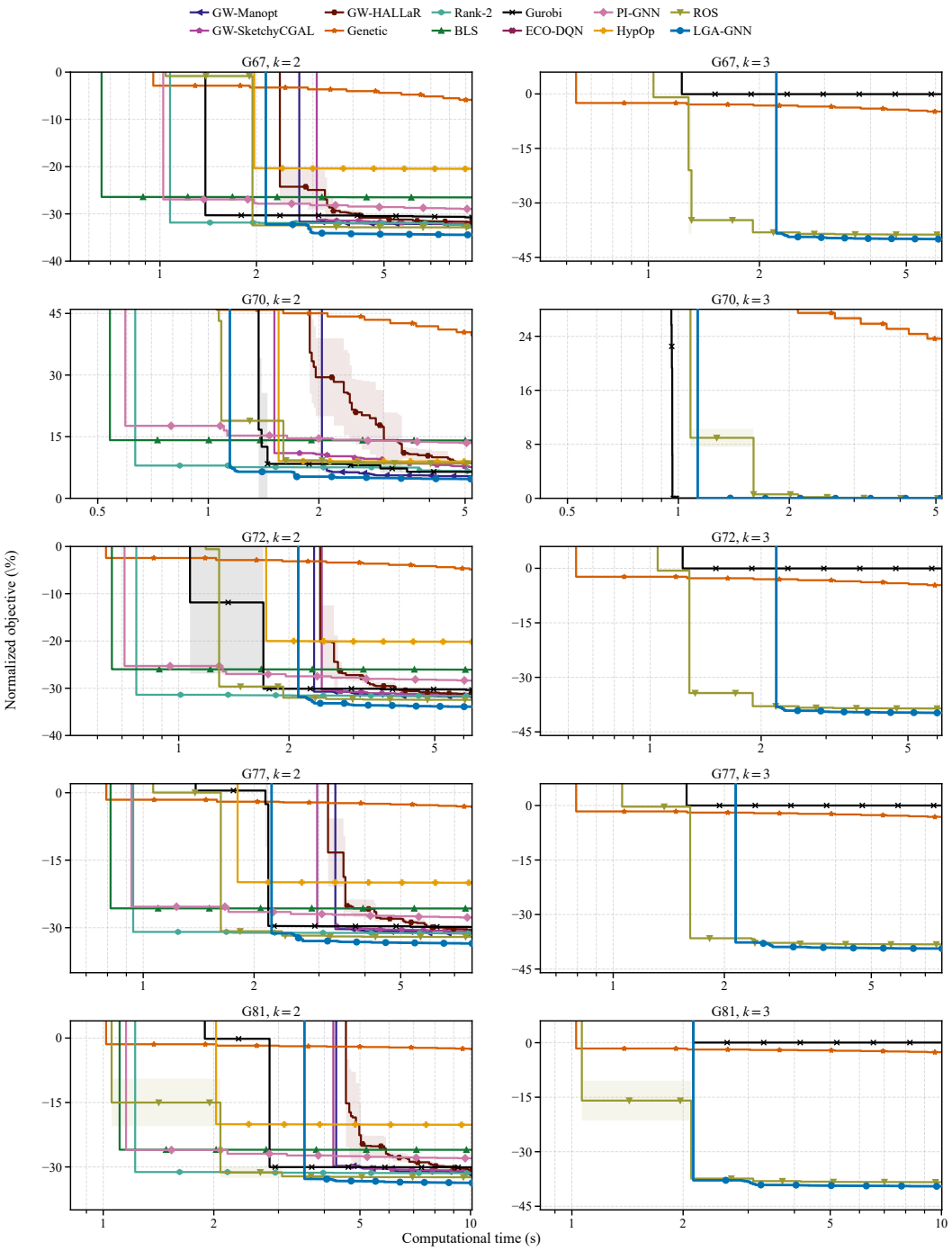

GW-Manopt
GW-SketchyCGAL
GW-HALLaR
Genetic
Rank-2
BLS
Gurobi
ECO-DQN
PI-GNN
HypOp
ROS
LGA-GNN
G37, k = 2
G37, k = 3
G38, k = 2
G38, k = 3
G39, k = 2
G39, k = 3
G40, k = 2
G40, k = 3
G41, k = 2
G41, k = 3
G42, k = 2
G42, k = 3
Normalized objective (%)
Computational time (s)

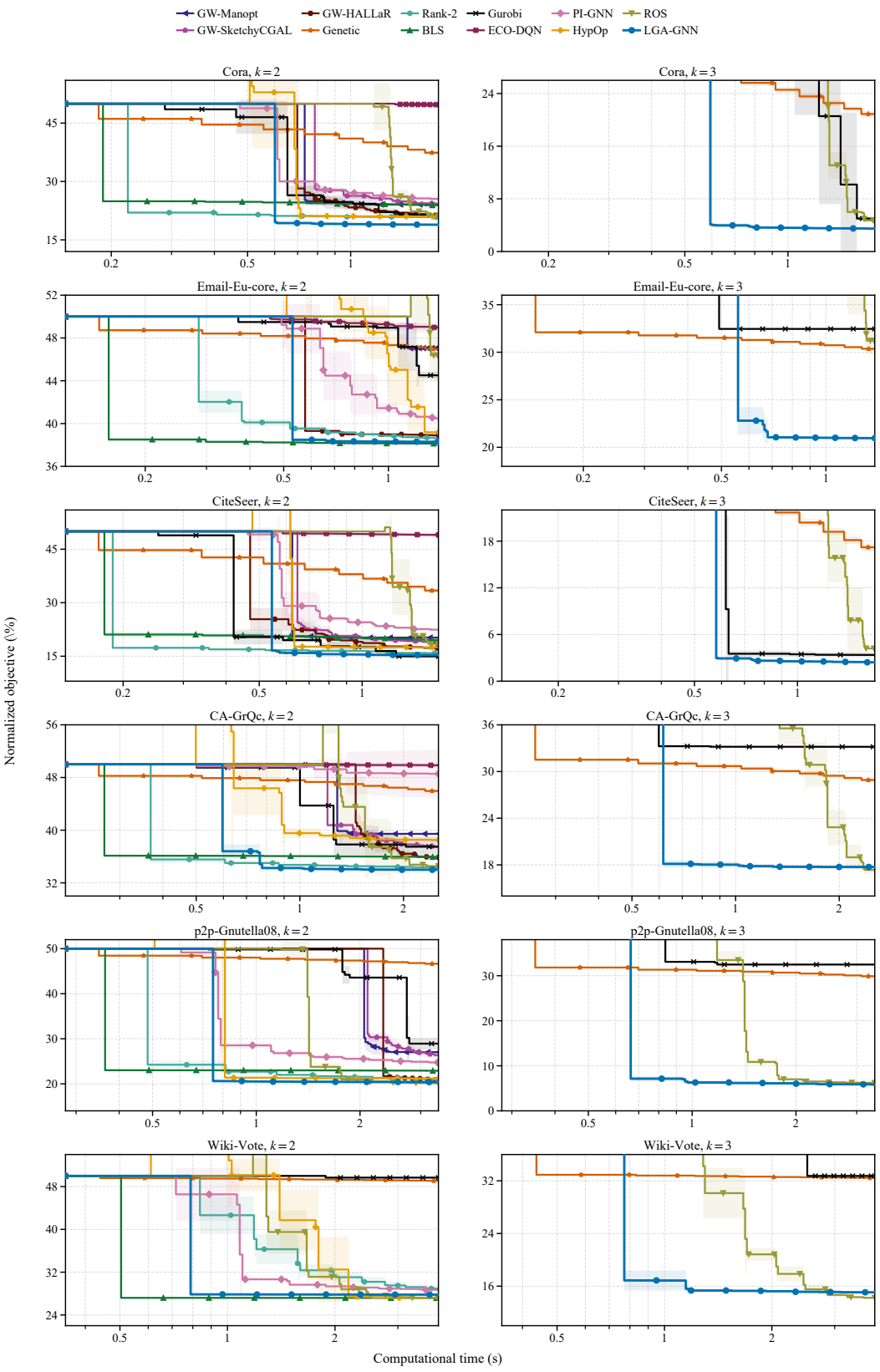

GW-Manopt
GW-SketchyCGAL
GW-HALLaR
Genetic
Rank-2
BLS
Gurobi
ECO-DQN
PI-GNN
HypOp
ROS
LGA-GNN
G43, k = 2
G43, k = 3
G44, k = 2
G44, k = 3
G45, k = 2
G45, k = 3
G46, k = 2
G46, k = 3
G47, k = 2
G47, k = 3
G48, k = 2
G48, k = 3
Normalized objective (%)
Computational time (s)


GW-Manopt
GW-SketchyCGAL
GW-HALLaR
Genetic
Rank-2
BLS
Gurobi
ECO-DQN
PI-GNN
HypOp
ROS
LGA-GNN
G49, k = 2
G49, k = 3
G50, k = 2
G50, k = 3
G51, k = 2
G51, k = 3
G52, k = 2
G52, k = 3
G53, k = 2
G53, k = 3
G54, k = 2
G54, k = 3
Normalized objective (%)
Computational time (s)


GW-Manopt
GW-SketchyCGAL
GW-HALLaR
Genetic
Rank-2
BLS
Gurobi
ECO-DQN
PI-GNN
HypOp
ROS
LGA-GNN
G55, k = 2
G55, k = 3
G56, k = 2
G56, k = 3
G57, k = 2
G57, k = 3
G58, k = 2
G58, k = 3
G59, k = 2
G59, k = 3
G60, k = 2
G60, k = 3
Normalized objective (%)
Computational time (s)


GW-Manopt
GW-SketchyCGAL
GW-HALLaR
Genetic
Rank-2
BLS
Gurobi
ECO-DQN
PI-GNN
HypOp
ROS
LGA-GNN
G61, k = 2
G61, k = 3
G62, k = 2
G62, k = 3
G63, k = 2
G63, k = 3
G64, k = 2
G64, k = 3
G65, k = 2
G65, k = 3
G66, k = 2
G66, k = 3
Normalized objective (%)
Computational time (s)


GW-Manopt
GW-SketchyCGAL
GW-HALLaR
Genetic
Rank-2
BLS
Gurobi
ECO-DQN
PI-GNN
HypOp
ROS
LGA-GNN
G67, k = 2
G67, k = 3
G70, k = 2
G70, k = 3
G72, k = 2
G72, k = 3
G77, k = 2
G77, k = 3
G81, k = 2
G81, k = 3
Normalized objective (%)
Computational time (s)


GW-Manopt
GW-SketchyCGAL
GW-HALLaR
Genetic
Rank-2
BLS
Gurobi
ECO-DQN
PI-GNN
HypOp
ROS
LGA-GNN
Cora, $k=2$
Cora, $k=3$
Email-Eu-core, $k=2$
Email-Eu-core, $k=3$
CiteSeer, $k=2$
CiteSeer, $k=3$
CA-GrQc, $k=2$
CA-GrQc, $k=3$
p2p-Gnutella08, $k=2$
p2p-Gnutella08, $k=3$
Wiki-Vote, $k=2$
Wiki-Vote, $k=3$
Normalized objective (%)
Computational time (s)