NEURAL ALGORITHMIC REASONING FOR GRAPH SADDLE-POINT PROBLEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural algorithmic reasoning, or aligning a neural network with an algorithmic paradigm, has emerged as an approach to solving polynomial-time-solvable and computationally harder combinatorial optimization problems. We propose a new message-passing framework based on the Chambolle-Pock Primal–Dual Hybrid Gradient (PDHG) method called GRAPHPDHG for solving general graph saddle-point problems. Theoretically, we show that GRAPHPDHG can efficiently solve a family of graph saddle-point problems by simulating PDHG. We also show that our network can learn an accelerated PDHG algorithm. Experimentally, we support our results on accelerated PDHG by evaluating the performance of our model as a learned warm start for second-order optimization techniques (SSNAL). We also show that alignment with PDHG leads to stronger size generalization than non-aligned graph neural network (GNN) baselines. Overall, we propose a novel architecture for solving a general family of optimization problems on graphs.

1 INTRODUCTION

Data-dependent methods to learn better and faster approximations to difficult combinatorial optimization problems have become increasingly popular and important for downstream applications like chip design, scheduling, matching, and related applications Bengio et al. (2021). Neural algorithmic reasoning (NAR), where neural networks are designed to imitate algorithmic control flows (Veličković et al., 2020; Velickovic & Blundell, 2021; Veličković et al., 2022), has emerged as a promising technique for improving the accuracy of such models. In the context of graph-structured problems, neural algorithmic reasoning has also been shown to improve out-of-distribution generalization, or the ability of a network to adapt to test instances that are sampled from a different distribution than the training instances (Cappart et al., 2023; Yang et al., 2021; Yau et al., 2024; Nerem et al., 2025). Neural algorithmic reasoning has been shown to be especially effective with respect to size generalization, when the size of the test input graph is larger than the size of the training graphs. By biasing the model toward an algorithm, one hopes that the algorithm will learn a generalizable procedure that will deliver robust results across different problem instances and additionally, avoid expensive handcrafted feature engineering.

In this paper, we leverage neural algorithmic reasoning to design efficient neural models for solving graph-structured (non)convex-(non)concave saddle-point problems, which appear across statistics, signal processing and machine learning (Chambolle & Pock, 2011; Tibshirani et al., 2005; Chen et al., 2014; Hein & Setzer, 2011). These problems include many standard graph-structured problems, including graph partitioning, convex clustering, and p-Wasserstein distance between graphs (Hein & Setzer, 2011; Ye et al., 2020; Jung et al., 2019; Chen et al., 2014; Chi & Lange, 2015; Peyré & Cuturi, 2020; Robertson et al., 2025a), and are most commonly adopted in problems where one must trade off between a reconstruction error and a regularity penalty under constraints. We are particularly motivated by the Primal–Dual Hybrid Gradient (PDHG) method (Chambolle & Pock, 2011; Pock & Chambolle, 2011). Classical PDHG is an *iterative* general-purpose solver, but its practical performance depends strongly on certain instance-dependent choices (e.g., step sizes, diagonal preconditioning, extrapolation/momentum, stopping criteria) (Applegate et al., 2021; Möllenhoff et al., 2018; Ye et al., 2020). In the context of graph saddle point problems, when the reconstruction error and regularity penalty admit certain separable structure and are "simple" in the sense of computing their associated proximal operators, PDHG is particularly efficient and amenable to message passing. In contrast to other first-order or second-order methods, such as ADMM (Chi

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.

& Lange, 2015) or semismooth Newton methods Sun et al. (2021), which require solving expensive global subproblems, PDHG can take advantage of the graph structure and only requires simple local message passing updates.

In this paper, we leverage this message passing structure to design a PDHG-inspired message passing neural network (MPNN). Importantly, our neural-PDHG architecture allows us to learn algorithmic parameters to the typical spectra/conditioning and feature statistics of the instances at hand. This is especially valuable when one must solve many related graph problems under tight compute budgets, when good preconditioners are expensive to design, or when one has additional supervision.

To design an efficient neural model to solve these graph saddle-point problems, we are motivated by the Primal–Dual Hybrid Gradient (PDHG) method (Chambolle & Pock, 2011; Pock & Chambolle, 2011). Classical PDHG is an *iterative* general-purpose solver, but its practical performance depends strongly on certain instance-dependent choices (e.g., step sizes, diagonal preconditioning, extrapolation/momentum, stopping criteria) (Applegate et al., 2021; Möllenhoff et al., 2018; Ye et al., 2020). In the context of graph saddle point problems, when the reconstruction error and regularity penalty admit certain separable structure and are "simple" in the sense of computing their associated proximal operators, PDHG is particularly efficient and amenable to message passing. In contrast to other first-order or second-order methods, such as ADMM (Chi & Lange, 2015) or semismooth Newton methods Sun et al. (2021), which require solving expensive global subproblems, PDHG can take advantage of the graph structure and only requires simple local message passing updates.

In this paper, we leverage this message passing structure to design a PDHG-inspired message passing neural network (MPNN). Importantly, our neural-PDHG architecture allows us to learn algorithmic parameters to the typical spectra/conditioning and feature statistics of the instances at hand. This is especially valuable when one must solve many related graph problems under tight compute budgets, when good preconditioners are expensive to design, or when one has additional supervision.

Contributions. We introduce GRAPHPDHG, a message passing GNN scheme inspired by PDHG which can be used to solve general graph saddle-point problems. In contrast to earlier work (Numeroso et al., 2023; He & Vitercik, 2025; Li et al., 2024) which targets algorithmic *execution* for discrete primal–dual schemes in neural algorithmic reasoning, we focus on *continuous* primal–dual optimization for graph-structured composite objectives defined on point clouds and graphs. Our contributions include:

- **A message passing PDHG network.** We introduce PDHG-inspired graph neural network for graph-based saddle-point problems with convex and nonconvex penalties, which we call GRAPHPDHG. GRAPHPDHG is designed to carefully exploit primal–dual saddle structure, while avoiding costly inner solves, and yielding simple, scalable iterations based on sparse graph operators and proximal updates. In Section 3, we introduce a node-edge message-passing layer that operates locally on a graph and is tailored to graph-regularized objectives $G(u) + F(Ku)$. We show in Section B.1 that a single layer of GRAPHPDHG can implement a single step of preconditioned PDHG. We apply our architecture to solving convex clustering.
- **GRAPHPDHG and learned acceleration.** In Section 3.1, we analyze the linearized primal–dual error dynamics induced by the layer and show that in a memoryless regime the update reduces to a preconditioned graph-Laplacian dynamics (Proposition 3.4). Alternatively, with edge memory, the recurrence recovers heavy-ball acceleration (Polyak, 1964) (Proposition 3.5).
- **Empirical performance.** We evaluate our model performance with respect to the convex clustering problem. We show that our PDHG-based model produces high-quality warm starts for both PDHG and second-order optimization methods. We further demonstrate that GRAPHPDHG generalizes better than GNN baselines to test graphs that are larger than those seen during training.

Related work. We give more exposition on the existing literature in the appendix and restrict our attention here to the most relevant prior work. There is a large body of prior work in understanding how neural networks solve a variety of optimization problems from both supervised Georgiev et al. (2024) and unsupervised approaches Karalias & Loukas (2020) to mixed neural-algorithmic pipelines Kahng et al. (2024); McCarty et al. (2021). The problems considered range from classical combinatorial optimization problems such as max-independent set and minimum dominating

set Loukas (2019); Sato et al. (2021) to semidefinite programs Yau et al. (2024). One of the earliest works to connect neural algorithmic reasoning with duality is Numeroso et al. (2023), where they develop dual algorithmic reasoning frameworks for solving min-cut/max-flow and shortest-path problems. More recently, PDNAR He & Vitercik (2025) and PDHG-net Li et al. (2024) develop neural frameworks aligned with primal–dual algorithms, focusing on minimum hitting set and linear programming problems, respectively. In contrast to previous work, our model is explicitly designed to handle graph-based optimization problems. We consider a broader class of problems which includes linear, nonlinear, and nonsmooth graph-structured saddle-point problems. Finally, we differ from existing algorithmic reasoning works as GRAPHPDHG is trained end-to-end to minimize the primal–dual gap in an *unsupervised manner*, without the need for intermediate supervision or step-by-step algorithmic guidance.

2 PRELIMINARIES

Graph neural networks. Graph neural networks (GNNs) are neural architectures for graph-structured data (Scarselli et al., 2009). Let $\mathcal{G} = (V, E, w)$ be a weighted graph with node features $X = \{x_i \in \mathbb{R}^d\}_{i \in V}$ and, optionally, edge features $e = \{e_{(i,j)}\}_{(i,j)\in E}$. Intuitively, a message-passing GNN computes new node features at each layer by aggregating information from its neighbors (Gilmer et al., 2017; Jegelka, 2022). To be exact, in a generic layer, the feature for a node i computed at each layer is

$$h_i^{(\ell+1)} = f_{\text{up}}^{(\ell)}\left(h_i^{(\ell)}, f_{\text{agg}}^{(\ell)}\left(\left\{h_j^{(\ell)}, e_{(i,j)}, w_{ij} : j \in \mathcal{N}(i)\right\}\right)\right), \tag{1}$$

where $\mathcal{N}(i)$ denotes the neighbors of i and $h_i^{(\ell)}$ is the learned node feature. Since the aggregation and update maps are shared across vertices, a GNN layer can be applied to graphs with different sizes and topologies.

Graph saddle-point problems. We consider problems of the form

$$\min_{u \in \mathbb{R}^{|V|\times d}} \{G(u) + F(Ku)\}, \tag{2}$$

where the model is formulated on an undirected graph $\mathcal{G} = (V, E, w)$ with positive weights $w : E \to \mathbb{R}_+$. G is commonly a function on vertices and F is a function on edges. In the canonical graph-structured setting, K is a weighted incidence operator. Fix an arbitrary orientation of each edge $e = (i, j) \in E$ and define $(Ku)_e = \sqrt{w_e}(u_i - u_j)$. Writing ∇ for the (signed) incidence matrix of $\mathcal{G}$, one can compactly express $K = \text{diag}(\sqrt{w})\nabla,\ \forall e = (i, j) \in E$ so that Ku collects weighted differences across edges.

An equivalent way to express eq. (2) is via its saddle-point (primal–dual) formulation. The Fenchel duality framework gives the equivalent saddle problem

$$\min_{u \in \mathbb{R}^{|V|\times d}} \max_{p \in \mathbb{R}^{|E|\times d}} \{G(u) + \langle Ku, p\rangle - F^*(p)\}, \tag{3}$$

where F^* denotes the convex conjugate of F, and $\langle Ku, p\rangle = \sum_{e\in E}\langle (Ku)_e, p_e\rangle$. Under standard qualification conditions, eq. (3) has the same optimal value as eq. (2), and saddle points $(u^\star, p^\star)$ satisfy the KKT system associated with the original problem.

Remark 2.1 (Primal–dual gap)**.** *Given the objective $f(u) = G(u) + F(Ku)$, the dual objective is $d(p) = -G^*(-K^\top p) - F^*(p)$, where G^* and F^* are convex conjugates. The primal–dual gap is*

$$\textit{Gap}(u, p) = f(u) - d(p) = G(u) + F(Ku) + G^*(-K^\top p) + F^*(p) \geq 0 \tag{4}$$

Under the strong duality condition, $\textit{Gap}(u^*, p^*) = 0$ iff u^* and p^* are the optimal primal and dual variables respectively. Otherwise, the gap is strictly positive and weak duality holds.

Convex clustering. Classical convex clustering (and similarly Network Lasso) of points $x_1, \ldots, x_n \in \mathbb{R}^d$ can be written as a special case of eq. (2). We illustrate this in the Appendix in Table 2. Let the graph $\mathcal{G} = (V, E, w)$ have vertices $V = \{1, \ldots, n\}$, edge weights $w_e > 0$ for $e = (i, j) \in E$. Define

$$G(u) = \frac{1}{2}\sum_{i\in V}\|u_i - x_i\|_2^2, \quad F(z) = \sum_{e\in E}\lambda_e\|z_e\|_2,$$

where $\lambda_e := \lambda\sqrt{w_e}$. Then eq. (2) becomes

$$\min_{u\in\mathbb{R}^{|V|\times d}} \frac{1}{2}\sum_{i\in V}\|u_i - x_i\|_2^2 + \lambda \sum_{(i,j)\in E} w_{ij}\|u_i - u_j\|_2, \tag{5}$$

See the Appendix for more details.

Primal–Dual Hybrid Gradient (PDHG). The preconditioned Primal–Dual Hybrid Gradient (PDHG) method is a popular framework for solving the problem in eq. (3). Generic PDHG iterations alternate between these two local operations. First, the current node variables are compared across each edge, and the corresponding edge dual variable is updated by a proximal or projection step. Second, each node aggregates the updated dual variables on its incident edges and uses this aggregate to update its own primal variable. Thus, PDHG has the same computational pattern as node–edge message passing: node states send differences to edges, and edge states send corrective messages back to nodes. More concretely, a general formulation of the PDHG iterations is given as follows:

$$u^{(k+1)} = \arg\min_{u\in\mathbb{R}^{|V|\times d}} G(u) + \langle p^{(k)}, Ku\rangle + \frac{s}{2}\|u - u^{(k)}\|_S^2 \tag{6}$$

$$p^{(k+1)} = \arg\min_{p\in\mathbb{R}^{|E|\times d}} F^*(p) - \langle K(2u^{(k+1)} - u^{(k)}),\ p\rangle + \frac{t}{2}\|p - p^{(k)}\|_T^2 \tag{7}$$

Here, $S \in \mathbb{R}^{|V|\times|V|}$ and $T \in \mathbb{R}^{|E|\times|E|}$ are symmetric positive definite matrices, $s, t > 0$ and the norm $\|\cdot\|_A$ for some matrix A of a vector u is defined by the quadratic $\|u\|_A^2 = u^\top Au$. For a given S and T, the convergence of PDHG is guaranteed if the (inverse) step sizes s and t satisfy $st > \|T^{-1/2}KS^{-1/2}\|^2$. One key feature of PDHG is that the updates described in eq. (6) and eq. (7) become fully local under certain separability assumptions on G and F and their associated proximal maps. As we will demonstrate, this observation is the main algorithmic reason for using PDHG as the template for GRAPHPDHG. While classical PDHG fixes the proximal maps, step sizes, preconditioners, and extrapolation rule, in GRAPHPDHG, we allow certain parts of the update to be learned, so that the network can adapt its effective preconditioning and acceleration to a family of graph topologies and feature statistics of a problem distribution.

Theorem 2.2 (Pock & Chambolle (2011))**.** *Assume G and F are proper, lower semicontinuous, convex, and that eq. (3) admits a saddle point $(u^\star, p^\star)$. Let $(u^{(k)}, p^{(k)})$ be generated by eq. (6) and eq. (7) such that*

$$st > \left\|T^{-1/2}KS^{-1/2}\right\|_2^2.$$

Let $\bar{u}^{(N)} := \frac{1}{N}\sum_{k=1}^N u^{(k)}$ and $\bar{p}^{(N)} := \frac{1}{N}\sum_{k=1}^N p^{(k)}$. Then for all $N \geq 1$,

$$\mathcal{L}(\bar{u}^{(N)}, p^\star) - \mathcal{L}(u^\star, \bar{p}^{(N)}) \leq \frac{1}{N}\left(\frac{s}{2}\|u^{(0)} - u^\star\|_S^2 + \frac{t}{2}\|p^{(0)} - p^\star\|_T^2\right),$$

where $\mathcal{L}(u, p) := G(u) + \langle Ku, p\rangle - F^*(p)$, i.e., the primal–dual gap converges as $\mathcal{O}(1/N)$.

The above theorem also motivates our unsupervised training objective. In particular, in Corollary B.1 in the Appendix, we show that when G is strongly convex, the primal–dual gap bounds the distance from the primal iterate to the unique primal optimizer. Thus, even without access to ground-truth solutions or intermediate PDHG iterates, minimizing the gap controls the iterate error. This is one important advantage of GRAPHPDHG over recent work in NAR, which generally require intermediate layer hints or algorithmic supervision.

3 MESSAGE PASSING PDHG

The PDHG formulation (eq. (3)) accommodates a broad range of edge penalties. In the separable setting, where $F(z) = \sum_{e\in E} F_e(z_e)$, the choice of F_e specifies the coupling prior across edges. In the Appendix, we list a few examples of separable convex and nonconvex edge penalties with simple proximal operators.

We now present our framework using a GNN to simulate the general primal–dual approximation algorithm, representing the primal–dual as node and edge features, respectively.

We adopt the encoder-processor-decoder architecture from Hamrick et al. (2018) implemented in the neural algorithmic reasoning pipeline Velickovic & Blundell (2021). In this framework, the *processor* is taken to be GRAPHPDHG, operating on the latent space learned by an *encoder*. The *decoder* reconstructs the final representation of the solution from the latent space. The framework is trained end-to-end to minimize the *primal–dual gap*. In this framework, the processor network is meant to execute a general numerical algorithm while the encoder and decoder are simple layers used to transform the data to and from the latent representations required by the processor module to natural outputs. Note that although only the processor part of the pipeline uses our GRAPHPDHG architecture, we still refer to the entire encoder-processor-decoder with a GRAPHPDHG processor as GRAPHPDHG.

Encoder. The encoder architecture of GRAPHPDHG consists of two MLP encoders, f_n and f_e, which encode the node and edge inputs. These encoders transform the feature inputs into a high-dimensional latent space for the processor:

$$x_0 = f_n(x), \quad e^{(0)} = f_e(e)$$

Processor. A single layer of the processor Φ_θ is defined by

$$(h^{(\ell+1)}, e^{(\ell+1)}) = \Phi_\theta(h^{(\ell)}, e^{(\ell)}; \mathcal{G}, X_0).$$

We define a message passing layer for PDHG as follows.

Let $\theta = (A_1, A_2, B_1, B_2, B_3)$ be a collection of learnable linear maps and φ a differentiable nonlinearity. Define

$$f_{\text{up}}^{\text{edge}}\left(e_{(i,j)}^{(\ell)}\right) = A_1 e_{(i,j)}^{(\ell)}, \quad f_{\text{agg}}^{\text{edge}}\left(h_i^{(\ell)} - h_j^{(\ell)}\right) = A_2\left(\sqrt{w_{ij}}(h_i^{(\ell)} - h_j^{(\ell)})\right),$$

$$f_{\text{up}}^{\text{node}}\left(h_j^{(\ell)}, \sum_{i \in N(j)} e_{(i,j)}^{(\ell+1)}\right) = \varphi\left(B_1 h_j^{(\ell)} + B_2(x_0)_j + B_3 \sum_{i \in N(j)} e_{(i,j)}^{(\ell+1)}\right),$$

$$\mathsf{NORMALIZE}(e) = \text{prox}_{\sigma F^*}(e).$$

One step of PDHG message passing is defined as

$$e_{(i,j)}^{(\ell+1)} = \mathsf{NORMALIZE}(f_{\text{up}}^{\text{edge}}(e_{(i,j)}^{\ell}) + f_{\text{agg}}^{\text{edge}}(h_i^{(\ell)} - h_j^{(\ell)}))$$
$$h_j^{(\ell+1)} = f_{\text{up}}^{\text{node}}(h_j^{(\ell)}, \sum_{i \in N(j)} e_{(i,j)}^{(\ell+1)}).$$

The L-layer processor is the composition of L PDHG message passing layers. $\mathcal{N}_\theta^L := \Phi_{\theta_L} \circ \ldots \circ \Phi_{\theta_1}$. The layers of the processor can be interpreted as a learnable version of one PDHG step. The matrix A_1 controls how much edge-dual information is retained from the previous layer. The matrix A_2 controls how node differences are transformed before updating the edge state. The matrices B_1, B_2, and B_3 combine the previous node state, the original input features, and the aggregated edge feedback. In classical PDHG, these transformations are fixed scalars (e.g., step sizes) or simple preconditioners; in GRAPHPDHG, they are learned from a distribution of training graph instances.

Decoder. As with the encoder, the architecture implements two decoders for edges and nodes: $\widehat{f}_n$, $\widehat{f}_e$ which map from the latent space to the data space. We show in the following theorem that this architecture is universal. If G decomposes over nodes, F decomposes over edges, and the associated proximal maps are separable, then for any desired accuracy, there exists a choice of parameters with depth bounded by a quantity depending on the spectral norm of the graph Laplacian such that the network achieves an arbitrarily small primal–dual gap. The proof is deferred to the appendix.

Theorem 3.1 (Universality)**.** *Let $\epsilon > 0$ be a target primal–dual gap. There exists a parameter configuration θ^* and a depth L satisfying*

$$L \leq \mathcal{O}\left(\frac{\|K\|^2}{\epsilon}\right)$$

such that the network achieves an ergodic primal–dual gap of at most ϵ. Here, $\|K\|$ is the spectral norm of the weighted incidence matrix.

3.1 FEEDFORWARD DYNAMICS OF GRAPHPDHG

We have shown that GRAPHPDHG is expressive enough to implement PDHG. In this section, we will show that our proposed architecture is also capable of learning an accelerated optimization algorithm as compared to vanilla PDHG. Our results here imply that for graphs that exhibit rapid eigenvalue decay, GRAPHPDHG can learn an accelerated algorithm. We consider the linearized feedforward dynamics of PDHG-inspired message passing. Specifically, we characterize these dynamics under the following set of assumptions:

Assumption 3.2. *The layer is operating in the Linear Regime if the following two conditions hold:*

1. *The activation function $\varphi_{node}(\cdot)$ is the identity.*
2. *For the projection $\text{Proj}_{\mathcal{C}}(P)$ onto the ℓ_2-ball constraints, there exists a partition of edges $E = E_{active} \cup E_{inactive}$. In the linear regime, the projection acts as the identity on $E_{inactive}$.*

In the classical analysis of nonlinear proximal methods, the iterates $(u^{(k)}, p^{(k)})$ converge in two stages. In the first stage, the set of "active" constraints that lie on the boundary of the feasible set $\mathcal{C}$ stabilizes Liang et al. (2018). Once this active set is identified, the projection operator $\text{Proj}_{\mathcal{C}}$ simplifies locally to an affine operator Hare & Lewis (2004). Namely, it acts as the identity for inactive variables and a projection onto the tangent space for active variables. Within this linear regime, the linearization of our framework is characterized by the following updates:

Remark 3.3. *Let $\mathcal{I} = \{e \in E : \|p_e\| < \lambda_e\}$. In the linear regime, the message passing updates are:*

$$e^{(\ell+1)} = e^{(\ell)} A_1^\top + K U^{(\ell)} A_2^\top$$
$$U^{(\ell+1)} = U^{(\ell)} B_1^\top + (K^\top e^{(\ell+1)}) B_3^\top + C$$

where A_1, A_2 and B_1, B_3 are the network parameters, and C is some constant. Substituting the edge update into the node update recovers the following recursion:

$$U^{(\ell+1)} = U^{(\ell)} B_1^\top + (K^\top e^{(\ell)}) A_1^\top B_3^\top + (K^\top K) U^{(\ell)} A_2^\top B_3^\top + C,$$

where $K^\top K$ is the graph Laplacian.

Analyzing the recursion yields a pair of results. The first result demonstrates that a certain parameterization implies fast convergence on expander-type graphs that exhibit rapid eigenvalue decay. In particular, the result below says that each layer can eliminate a source of error associated with one eigen-direction of the graph Laplacian.

Proposition 3.4 (Memoryless dynamics)**.** *Let $\Delta = K^\top K \in \mathbb{R}^{|V| \times |V|}$ be the graph Laplacian restricted to the inactive edge set defined in the Linear Regime. Let $\{(\lambda_k, \psi_k)\}_{k=1}^{|V|}$ be the orthonormal eigenpairs of Δ. Define the error matrix $\mathcal{E}^{(\ell)} = U^{(\ell)} - U^*$.*

For any target eigenvalue $\lambda_k \neq 0$, there exists a parameterization of the GNN layer such that the error at the next layer, $\mathcal{E}^{(\ell+1)}$, satisfies $\psi_k^\top \mathcal{E}^{(\ell+1)} = 0$. That is, the projection of the error onto the k-th eigenspace is eliminated for all feature dimensions.

While this result implies the algorithm learned by GRAPHPDHG can rapidly converge on graphs with fast eigenvalue decay, the implied theoretical convergence rate is still $O(\kappa)$. Below, we demonstrate that the network has the capacity to learn faster, accelerated algorithms that converge in $O(\sqrt{\kappa})$. In particular, retaining edge memory yields a second-order recurrence which bears a similarity to classical accelerated first-order methods.

Proposition 3.5 (Dynamics with memory)**.** *Let the GNN operate in the linear regime with edge memory $A_1 = \gamma I_d$ for some $\gamma \in (0, 1)$. The primal error dynamics satisfy the second-order recurrence of the Polyak Heavy Ball method Polyak (1964):*

$$\mathcal{E}^{(\ell+1)} = \mathcal{E}^{(\ell)} + \gamma(\mathcal{E}^{(\ell)} - \mathcal{E}^{(\ell-1)}) - \Delta \mathcal{E}^{(\ell)} M,$$

where M is a learnable effective step size matrix.

Corollary 3.6. *Consider the simplification $M = -A_2^\top B_3^\top = \eta I_d$ for $\eta > 0$. Then, the worst-case error decays at the accelerated rate $\max_k \|c_k^{(\ell)}\|_2 \leq C \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^\ell$ for some constant C.*

The results above imply convergence that is *independent of the size of the graph* for families of graphs with bounded condition number κ. This is a unique property of GRAPHPDHG and is a notable shortcoming of prior NAR works Loukas (2019). We improve the rate in Theorem B.3.

Corollary 3.7. *Let the GNN operate in the linear regime and let the primal error dynamics satisfy the second-order recurrence of the Polyak Heavy Ball method as in Theorem 3.5. Assume that at the first layer, the solution found by the neural network incurs some error c for each vertex. For graphs with bounded condition number κ, there is a parameter configuration for a depth $\mathcal{O}(\kappa)$ GNN such that $\mathcal{E} < \mathcal{O}(\kappa, 1/\varepsilon)$.*

3.2 PDHG FOR CONVEX CLUSTERING

Here, we use convex clustering (eq. (5)) as a case study for how our GRAPHPDHG framework can be directly applied. As previously noted, convex clustering is an important continuous relaxation of the clustering objective where one must balance between a reconstruction objective and a fusion penalty. This makes it a natural choice for PDHG to solve. We derive the explicit message passing form of PDHG for convex clustering. This message passing form can then be directly used in our GRAPHPDHG framework to derive a neural version of PDHG for convex clustering.

For the choice of F implied by the convex clustering objective in eq. (5), the convex conjugate is

$$F^*(p) = \sum_{e \in E} \mathbb{I}_{\{\|p_e\|_2 \le \lambda_e\}}(p_e) = \mathbb{I}_{\mathcal{C}}(p), \quad \mathcal{C} := \{p : \|p_e\|_2 \le \lambda_e \ \forall e \in E\}.$$

For the quadratic term, G, the convex conjugate, used in eq. (4), is

$$G^*(q) = \sum_{i \in V} \left(\langle q_i, x_i \rangle + \tfrac{1}{2}\|q_i\|_2^2 \right) = \langle q, x \rangle + \tfrac{1}{2}\|q\|_2^2.$$

We now specialize the preconditioned PDHG iterations to this problem. Let $S \in \mathbb{R}^{|V|\times|V|}$ and $T \in \mathbb{R}^{|E|\times|E|}$ be symmetric positive definite, and let $s, t > 0$ be the (inverse) step sizes. Recall the generic PDHG updates eq. (6) and eq. (7). To obtain local message passing updates, we choose node- and edge-wise preconditioners $S = \mathrm{diag}(S_i)_{i\in V}$ and $T = \mathrm{diag}(T_e)_{e\in E}$, with $S_i \in \mathbb{R}^{d\times d}$, $T_e \in \mathbb{R}^{d\times d}$ symmetric positive definite blocks. Then the proximal operator (eq. (7)) decouples over edges and simplifies, for each $e \in E$:

$$p_e^{(k+1)} = \mathrm{Proj}_{\{\|p\|_2 \le \lambda_e\}} \left(p_e^{(k)} + \frac{1}{t} T_e^{-1} (K u^{\mathrm{extr}})_e \right), \tag{8}$$

where u^{extr} denotes the extrapolated primal variable (e.g., $2u^{(k)} - u^{(k-1)}$), and $(Ku^{\mathrm{extr}})_e = \sqrt{w_e}(u_i^{\mathrm{extr}} - u_j^{\mathrm{extr}})$ for $e = (i, j)$.

Similarly, eq. (6) decouples over vertices. The first-order optimality condition for each $i \in V$ yields

$$(I_d + sS_i)u_i^{(k+1)} = sS_i u_i^{(k)} + x_i - (K^\top p^{(k+1)})_i, \tag{9}$$

where $(K^\top p)_i = \sum_{e=(i,j)\in E} \sqrt{w_e} p_e$. Thus each primal update is a local linear transform of the previous iterate, the data x_i, and the aggregated dual variables on incident edges.

Remark 3.8. *A particularly simple choice is $S = I, \quad T = I, s = \frac{1}{\sigma}, t = \frac{1}{\tau}$, for step sizes $\sigma, \tau > 0$ satisfying $\tau\sigma\|K\|^2 < 1$. In this case, eq. (8) and eq. (9) simplify to the explicit message passing*

$$p_{ij}^{(k+1)} = \mathrm{Proj}_{\{\|p\|_2 \le \lambda_e\}} \left(p_{ij}^{(k)} + \tau\sqrt{w_{ij}} \left(u_i^{\mathrm{extr}} - u_j^{\mathrm{extr}} \right) \right), \tag{10}$$

$$u_i^{(k+1)} = \frac{1}{1+\sigma} \left(u_i^{(k)} + \sigma x_i - \sigma \sum_{j:(i,j)\in E} \sqrt{w_{ij}} p_{ij}^{(k+1)} \right). \tag{11}$$

4 EXPERIMENTAL RESULTS

In our experiments, we focus on two main questions: Can a PDHG-aligned model provide effective learned preconditioners or warm starts for generic optimization algorithms? Does alignment

with Primal–Dual Hybrid Gradient (PDHG) promote size generalization? Warm-starting is especially important for graph-structured optimization problems, as classical solvers such as PDHG and semismooth Newton methods exhibit runtime that depends nontrivially on the quality of the initial primal and dual variables. We use GRAPHPDHG to amortize high-quality initializations over a distribution of related graphs by producing primal–dual variables that are already near the relevant saddle point or have approximately identified the correct fused-edge structure. In our experiments, we demonstrate GRAPHPDHG's utility for convex clustering (see Section 3.2). See our appendix for all experimental details including experiments on other saddle-point problems.

To evaluate GRAPHPDHG on multi-scale geometric graphs, we use a synthetic hierarchical Gaussian cluster dataset, and three image datasets. The hierarchical Gaussian cluster dataset consists of graphs whose node features are sampled from a recursive Gaussian hierarchy. For the image datasets, we consider k-nearest neighbor graphs generated using features from MNIST (Deng, 2012), Fashion-MNIST (Xiao et al., 2017), and CIFAR-10 (Krizhevsky & Hinton, 2009).

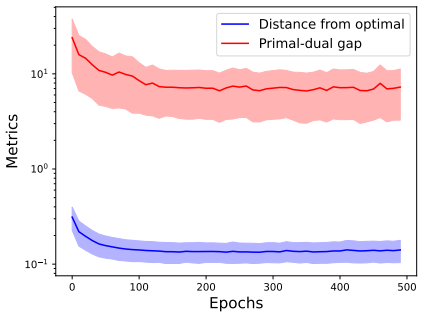

Figure 1: Tracking the primal–dual gap and the distance between the predicted and optimal primal variables throughout training. As the gap shrinks, so too does the iterate error.

We train GRAPHPDHG using the primal–dual gap (eq. (4)). By Theorem B.1, the primal–dual gap is an upper bound on the distance to the optimal solution, so a small gap implies that the predicted primal variables are close to optimal. Minimizing the primal–dual gap allows us to train the model in an unsupervised manner without requiring exact solutions or step-by-step algorithmic supervision. We observe that this theoretical relationship is reflected empirically. See Figure 1. We also note that our theoretical results assume *ergodic* iterates. In our experiments we use the last layer readouts.

We compare against several standard message-passing GNN baselines: GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018), GIN (Xu et al., 2019), GraphSAGE (Hamilton et al., 2017), and EdgeConv (Wang et al., 2019). For each baseline, we keep the same encoder–processor–decoder template and replace only the GRAPHPDHG processor with the corresponding GNN processor. All models were trained using the same primal–dual gap objective and experimental protocol.

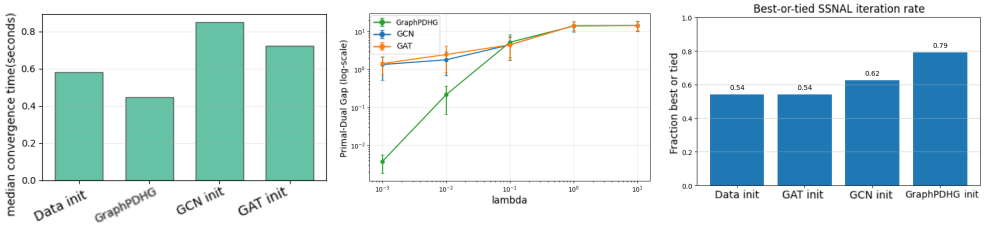

Figure 2: Warm-starting SSNAL for the hierarchical Gaussian dataset. **Left:** Time (seconds) to recover a primal–dual gap $\leq 10^{-6}$. **Middle:** The primal–dual gap of different network initializations as λ increases. **Right:** For each initialization method, the fraction of validation graphs for which SSNAL reaches a primal–dual gap of $\leq 10^{-6}$ at least as fast as the best-performing method.

Initializing a High-Accuracy Solver with GRAPHPDHG. We evaluate GRAPHPDHG as a learned warm start for the semismooth Newton augmented Lagrangian method (SSNAL), a state-of-the-art second-order solver for convex clustering (Sun et al., 2021). Rather than replacing SSNAL, GRAPHPDHG provides initial primal–dual variables that are passed to the classical solver. We measure whether this initialization reduces the iterations required to reach a target primal–dual gap, indicating that the network has learned useful problem structure such as the approximate fused or active dual-edge variables. We note that although GRAPHPDHG is aligned with PDHG, SSNAL is a robust high-accuracy solver whose performance benefits from an initialization that is close to the optimal primal–dual solution. Thus, we focus our experiments on SSNAL.

The results are summarized for the hierarchical Gaussian dataset in Figure 2. GRAPHPDHG provides a better warm-start than both simply initializing from the data or initializing from the baseline

Table 1: First-hit times for different initialization methods. Entries report mean ± standard deviation over runs. The best result is bold. Median hit times and ablations are reported in the Appendix.

		$\lambda = 2.5$			$\lambda = 10$		
Dataset	**Method**	10^{-4}	10^{-6}	10^{-8}	10^{-4}	10^{-6}	10^{-8}
MNIST	Data	2.0 ± 0.0	11.6 ± 4.8	11.6 ± 4.8	2.0 ± 0.0	6.4 ± 2.6	6.4 ± 2.6
	GIN	2.0 ± 0.0	12.0 ± 5.8	12.0 ± 5.8	2.2 ± 0.3	8.7 ± 4.8	8.7 ± 4.8
	GCN	2.0 ± 0.0	13.4 ± 6.8	13.4 ± 6.8	2.2 ± 0.2	8.5 ± 4.7	8.5 ± 4.7
	GAT	2.0 ± 0.0	7.0 ± 3.4	7.0 ± 3.4	2.4 ± 0.2	9.1 ± 5.1	9.1 ± 5.1
	GraphSAGE	2.0 ± 0.0	8.8 ± 4.5	8.8 ± 4.5	2.2 ± 0.2	7.4 ± 3.8	7.4 ± 3.8
	EdgeConv	2.0 ± 0.0	6.4 ± 3.0	6.4 ± 3.0	2.1 ± 0.2	6.4 ± 2.9	6.4 ± 2.9
	GRAPHPDHG	2.0 ± 0.0	$\mathbf{4.7 \pm 2.1}$	$\mathbf{4.7 \pm 2.1}$	$\mathbf{2.0 \pm 0.0}$	$\mathbf{5.6 \pm 2.3}$	$\mathbf{5.6 \pm 2.3}$
Fashion-MNIST	Data	5.4 ± 0.4	12.3 ± 4.9	12.3 ± 4.9	5.2 ± 0.4	11.4 ± 4.4	11.4 ± 4.4
	GIN	5.5 ± 0.6	14.5 ± 6.8	14.5 ± 6.8	5.3 ± 0.7	14.0 ± 6.8	14.0 ± 6.8
	GCN	5.4 ± 0.6	13.6 ± 6.5	13.6 ± 6.5	5.2 ± 0.6	13.1 ± 6.3	13.1 ± 6.3
	GAT	6.7 ± 1.2	17.8 ± 8.1	17.8 ± 8.1	6.4 ± 1.3	21.3 ± 9.9	21.3 ± 9.9
	GraphSAGE	5.3 ± 0.5	12.0 ± 5.2	12.0 ± 5.2	5.1 ± 0.5	11.8 ± 5.0	11.8 ± 5.0
	EdgeConv	5.2 ± 0.5	10.8 ± 4.7	10.8 ± 4.7	5.0 ± 0.5	10.6 ± 4.8	10.6 ± 4.8
	GRAPHPDHG	$\mathbf{5.1 \pm 0.4}$	$\mathbf{9.9 \pm 4.3}$	$\mathbf{9.9 \pm 4.3}$	$\mathbf{4.9 \pm 0.4}$	$\mathbf{9.4 \pm 4.2}$	$\mathbf{9.4 \pm 4.2}$
CIFAR-10	Data	$\mathbf{2.3 \pm 0.4}$	5.8 ± 2.4	5.8 ± 2.4	2.2 ± 0.4	5.4 ± 2.1	5.4 ± 2.1
	GIN	2.5 ± 0.6	6.8 ± 4.0	6.8 ± 4.0	2.4 ± 0.6	6.5 ± 3.7	6.5 ± 3.7
	GCN	2.5 ± 0.6	6.3 ± 3.7	6.3 ± 3.7	2.4 ± 0.6	6.1 ± 3.4	6.1 ± 3.4
	GAT	2.5 ± 0.6	12.5 ± 6.6	12.5 ± 6.6	2.4 ± 0.6	15.0 ± 7.3	15.0 ± 7.3
	GraphSAGE	2.4 ± 0.5	6.0 ± 3.0	6.0 ± 3.0	2.3 ± 0.5	5.9 ± 2.9	5.9 ± 2.9
	EdgeConv	$\mathbf{2.3 \pm 0.4}$	5.5 ± 2.3	5.5 ± 2.3	2.2 ± 0.4	5.2 ± 2.2	5.2 ± 2.2
	GRAPHPDHG	$\mathbf{2.3 \pm 0.4}$	$\mathbf{5.1 \pm 1.9}$	$\mathbf{5.1 \pm 1.9}$	$\mathbf{2.1 \pm 0.4}$	$\mathbf{4.8 \pm 1.8}$	$\mathbf{4.8 \pm 1.8}$

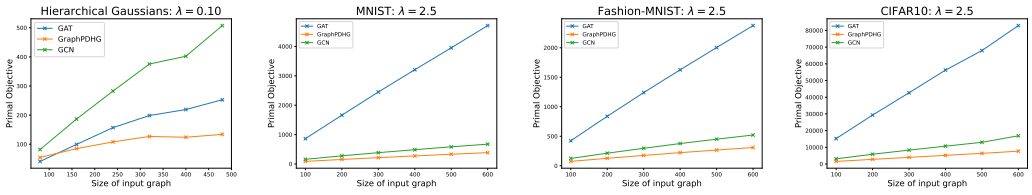

Figure 3: Size generalization for each dataset. We track the primal objective as the size of the test graphs increases for all datasets. Note that we test at a fixed λ for all graphs.

GNN outputs. The performance converges at higher λ as all methods snap to the trivial solution (i.e., the grand mean in the center). The results for the image datasets are shown in Table 1 where we record the number of iterations needed to reach different target primal–dual gap values. We show GRAPHPDHG initialization reduces the number of iterations SSNAL needs to converge.

Size generalization. We evaluate the size generalization of all neural network methods by measuring the primal objective value of the predicted initialization from each network on increasingly large graphs. The results are summarized in Figure 3 across all datasets. For all datasets, GRAPHPDHG achieves substantially lower primal objective values than the GAT and GCN baselines as the graph size increases. This suggests that GRAPHPDHG generalizes more effectively to larger problem instances. We attribute this behavior to the alignment of GRAPHPDHG with PDHG. By explicitly mirroring the structure of PDHG, GRAPHPDHG appears to learn a general rule for initialization that transfers more robustly than GAT or GCN across graph sizes.

5 CONCLUSION

In this paper, we introduced GRAPHPDHG, a primal–dual message-passing framework for graph-structured saddle-point problems. The central observation is that PDHG has the same computational structure as node-edge message passing. Theoretically, we showed that the proposed architecture can exactly represent preconditioned PDHG updates and that its linearized dynamics admit interpretations in terms of the Laplacian spectrum and acceleration. Empirically, we demonstrated the effectiveness of GRAPHPDHG on convex clustering problems, where the model improves size generalization and provides useful warm starts for classical solvers. These results suggest that learned primal–dual message passing can serve not only as a standalone approximation method, but also as a mechanism to accelerate classical optimization algorithms.

AI USE STATEMENT

In this work, we used generative AI tools for editing and writing tasks. Additionally, while the experimental pipeline was written by the authors without the use of AI tools, we used generative AI tools running experiments and writing code to create figures. We have reviewed all AI-assisted work. All LLM-generated code was verified and tested for correctness by two authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

In order to ensure reproducibility, all code and experimental configurations are provided is the supplemental materials.

REFERENCES

David Applegate, Mateo Diaz, Oliver Hinder, Haihao Lu, Miles Lubin, Brendan O' Donoghue, and Warren Schudy. Practical large-scale linear programming using primal-dual hybrid gradient. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 20243–20257. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/a8fbbd3b11424ce032ba813493d95ad7-Paper.pdf.

Ravi B Bapat and Stef Tijs. Incidence matrix games. *Game Theoretical Applications to Economics and Operations Research (Bangalore, 1996), Theory Decis. Lib. Ser. C, Game Theory Math. Program. Oper. Res*, 18:9–16, 1996.

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. doi: 10.1016/j.ejor.2020.07.063.

Patrick Breheny and Jian Huang. Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection. *The Annals of Applied Statistics*, 2011.

Quentin Cappart, Didier Chételat, Elias B Khalil, Andrea Lodi, Christopher Morris, and Petar Veličković. Combinatorial optimization and reasoning with graph neural networks. *Journal of Machine Learning Research*, 24(130):1–61, 2023.

Antonin Chambolle and Thomas Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145, 2011.

Gary Chen, Eric Chi, John Ranola, and Kenneth Lange. Convex clustering: An attractive alternative to hierarchical clustering. *PLoS computational biology*, 11, 09 2014. doi: 10.1371/journal.pcbi.1004228.

Gary K. Chen, Eric C. Chi, John Michael O. Ranola, and Kenneth Lange. Convex clustering: An attractive alternative to hierarchical clustering. *PloS: Computational Biology*, 2015.

Tianlong Chen, Xiaohan Chen, Wuyang Chen, Howard Heaton, Jialin Liu, Zhangyang Wang, and Wotao Yin. Learning to optimize: A primer and a benchmark. *Journal of Machine Learning Research*, 23(189):1–59, 2022.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1597–1607. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/chen20j.html.

Eric C Chi and Kenneth Lange. Splitting methods for convex clustering. *Journal of Computational and Graphical Statistics*, 24(4):994–1013, 2015.

Eric C. Chi, Aaron J. Molstad, Zheming Gao, and Jocelyn T. Chi. The why and how of convex clustering, 2025. URL https://arxiv.org/abs/2507.09077.

Li Deng. The MNIST database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Jianqing Fan and Runze Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, 2001.

Dobrik Georgiev Georgiev, Danilo Numeroso, Davide Bacciu, and Pietro Liò. Neural algorithmic reasoning for combinatorial optimisation. In *Learning on Graphs Conference*, pp. 28–1. PMLR, 2024.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1263–1272. PMLR, 2017. URL https://proceedings.mlr.press/v70/gilmer17a.html.

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.

David Hallac, Jure Leskovec, and Stephen Boyd. Network lasso: Clustering and optimization in large graphs. In *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Jessica B Hamrick, Kelsey R Allen, Victor Bapst, Tina Zhu, Kevin R McKee, Joshua B Tenenbaum, and Peter W Battaglia. Relational inductive bias for physical construction in humans and machines. *arXiv preprint arXiv:1806.01203*, 2018.

Jonas Berg Hansen and Filippo Maria Bianchi. Total variation graph neural networks. In *International Conference on Machine Learning*, pp. 12445–12468. PMLR, 2023.

Warren L Hare and Adrian S Lewis. Identifying active constraints via partial smoothness and prox-regularity. *Journal of Convex Analysis*, 11(2):251–266, 2004.

Yu He and Ellen Vitercik. Primal-dual neural algorithmic reasoning. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 22743–22766. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/he25r.html.

Matthias Hein and Simon Setzer. Beyond spectral clustering - tight relaxations of balanced graph cuts. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011. URL https://proceedings.neurips.cc/paper_files/paper/2011/file/193002e668758ea9762904da1a22337c-Paper.pdf.

Qiang Heng, Xiaoqian Liu, and Eric C. Chi. Anderson accelerated operator splitting methods for convex-nonconvex regularized problems. *IEEE Open Journal of Signal Processing*, 6:1094–1108, 2025. doi: 10.1109/OJSP.2025.3618583.

Stefanie Jegelka. Theory of graph neural networks: Representation and learning. In *Proceedings of the International Congress of Mathematicians*, 2022. URL https://arxiv.org/abs/2204.07697. Also available as arXiv:2204.07697.

Alexander Jung, Nguyen Tran, and Alexandru Mara. When is network lasso accurate? *Frontiers in Applied Mathematics and Statistics*, Volume 3 - 2017, 2018. ISSN 2297-4687. doi: 10.3389/fams.2017.00028. URL https://www.frontiersin.org/journals/applied-mathematics-and-statistics/articles/10.3389/fams.2017.00028.

Alexander Jung, Alfred O. Hero, III, Alexandru Cristian Mara, Saeed Jahromi, Ayelet Heimowitz, and Yonina C. Eldar. Semi-supervised learning in network-structured data via total variation minimization. *IEEE Transactions on Signal Processing*, 67(24):6256–6269, 2019. doi: 10.1109/TSP.2019.2953593.

Andrew B Kahng, Robert R Nerem, Yusu Wang, and Chien-Yi Yang. Nn-steiner: A mixed neural-algorithmic approach for the rectilinear steiner minimum tree problem. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 13022–13030, 2024.

Nikolaos Karalias and Andreas Loukas. Erdos goes neural: an unsupervised learning framework for combinatorial optimization on graphs. *Advances in Neural Information Processing Systems*, 33: 6659–6672, 2020.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013. URL https://api.semanticscholar.org/CorpusID:216078090.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. (0), 2009.

Bingheng Li, Linxin Yang, Yupeng Chen, Senmiao Wang, Qian Chen, Haitao Mao, Yao Ma, Akang Wang, Tian Ding, Jiliang Tang, and Ruoyu Sun. Pdhg-unrolled learning-to-optimize method for large-scale linear programming, 2024. URL https://arxiv.org/abs/2406.01908.

Jingwei Liang, Jalal Fadili, and Gabriel Peyré. Local linear convergence analysis of primal–dual splitting methods. *Optimization*, 67(6):821–853, 2018. doi: 10.1080/02331934.2018.1426584. URL https://doi.org/10.1080/02331934.2018.1426584.

Po-Ling Loh and Martin J. Wainwright. Regularized m-estimators with nonconvexity: Statistical and algorithmic theory for local optima. *Journal of Machine Learning Research*, 2015.

Po-Ling Loh and Martin J. Wainwright. Support recovery without incoherence: A case for nonconvex regularization. *The Annals of Statistics*, 2017.

Andreas Loukas. What graph neural networks cannot learn: depth vs width. *arXiv preprint arXiv:1907.03199*, 2019.

Evan McCarty, Qi Zhao, Anastasios Sidiropoulos, and Yusu Wang. Nn-baker: A neural-network infused algorithmic framework for optimization problems on geometric intersection graphs. *Advances in Neural Information Processing Systems*, 34:23023–23035, 2021.

Thomas Möllenhoff, Zhenzhang Ye, Tao Wu, and Daniel Cremers. Combinatorial preconditioners for proximal algorithms on graphs. In Amos Storkey and Fernando Perez-Cruz (eds.), *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pp. 38–47. PMLR, 09–11 Apr 2018. URL https://proceedings.mlr.press/v84/mollenhoff18a.html.

Robert R. Nerem, Samantha Chen, Sanjoy Dasgupta, and Yusu Wang. Graph neural networks extrapolate out-of-distribution for shortest paths, 2025. URL https://arxiv.org/abs/2503.19173.

Danilo Numeroso, Davide Bacciu, and Petar Velickovic. Dual algorithmic reasoning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=hhvkdRdWt1F.

Neal Parikh and Stephen Boyd. Proximal algorithms. 1(3):127–239, January 2014. ISSN 2167-3888. doi: 10.1561/2400000003. URL https://doi.org/10.1561/2400000003.

Gabriel Peyré and Marco Cuturi. Computational optimal transport, 2020. URL https://arxiv.org/abs/1803.00567.

Jiaxing Pi, Honggang Wang, and Panos M Pardalos. A dual reformulation and solution framework for regularized convex clustering problems. *European Journal of Operational Research*, 290(3): 844–856, 2021.

Thomas Pock and Antonin Chambolle. Diagonal preconditioning for first order primal-dual algorithms in convex optimization. In *2011 International Conference on Computer Vision (ICCV)*, pp. 1762–1769. IEEE, 2011. doi: 10.1109/ICCV.2011.6126441.

Boris T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 1964.

Sawyer Robertson, Zhengchao Wan, and Alexander Cloninger. All you need is resistance: On the equivalence of effective resistance and certain optimal transport problems on graphs, 2025a. URL https://arxiv.org/abs/2404.15261.

Sawyer Jack Robertson, Chester Holtz, Zhengchao Wan, Gal Mishne, and Alexander Cloninger. Robust graph-based semi-supervised learning via p-conductances, 2025b. URL https://arxiv.org/abs/2502.08873.

Rajiv Sambharya, Georgina Hall, Brandon Amos, and Bartolomeo Stellato. End-to-end learning to warm-start for real-time quadratic optimization. In Nikolai Matni, Manfred Morari, and George J. Pappas (eds.), *Proceedings of The 5th Annual Learning for Dynamics and Control Conference*, volume 211 of *Proceedings of Machine Learning Research*, pp. 220–234. PMLR, 15–16 Jun 2023. URL https://proceedings.mlr.press/v211/sambharya23a.html.

Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Random features strengthen graph neural networks. In *Proceedings of the 2021 SIAM international conference on data mining (SDM)*, pp. 333–341. SIAM, 2021.

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

Defeng Sun, Kim-Chuan Toh, and Yancheng Yuan. Convex clustering: Model, theoretical guarantee and efficient algorithm. *Journal of Machine Learning Research*, 22(9):1–32, 2021.

Robert Tibshirani, Michael Saunders, Saharon Rosset, Ji Zhu, and Keith Knight. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 67(1):91–108, 2005.

Petar Velickovic and Charles Blundell. Neural algorithmic reasoning. *CoRR*, abs/2105.02761, 2021. URL https://arxiv.org/abs/2105.02761.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Petar Veličković, Adrià Puigdomènech Badia, David Budden, Razvan Pascanu, Andrea Banino, Misha Dashevskiy, Raia Hadsell, and Charles Blundell. The clrs algorithmic reasoning benchmark. In *International Conference on Machine Learning*, pp. 22084–22102. PMLR, 2022.

Petar Veličković, Rex Ying, Matilde Padovano, Raia Hadsell, and Charles Blundell. Neural execution of graph algorithms. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SkgKO0EtvS.

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (tog)*, 38(5): 1–12, 2019.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *ArXiv*, abs/1708.07747, 2017.

Xingyu Xie, Jianlong Wu, Guangcan Liu, Zhisheng Zhong, and Zhouchen Lin. Differentiable linearized ADMM. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6902–6911. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/xie19c.html.

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

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., 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/file/1679091c5a880faf6fb5e6087eb1b2dc-Paper.pdf.

Yongyi Yang, Tang Liu, Yangkun Wang, Jinjing Zhou, Quan Gan, Zhewei Wei, Zheng Zhang, Zengfeng Huang, and David Wipf. Graph neural networks inspired by classical iterative algorithms. In *International Conference on Machine Learning*, pp. 11773–11783. PMLR, 2021.

Morris Yau, Nikolaos Karalias, Eric Lu, Jessica Xu, and Stefanie Jegelka. Are graph neural networks optimal approximation algorithms? In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 73124–73181. Curran Associates, Inc., 2024. doi: 10.52202/079017-2328. URL https://openreview.net/forum?id=SxRblm9aMs.

Zhenzhang Ye, Thomas Möllenhoff, Tao Wu, and Daniel Cremers. Optimization of graph total variation via active-set-based combinatorial reconditioning. In Silvia Chiappa and Roberto Calandra (eds.), *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 657–668. PMLR, 26–28 Aug 2020. URL https://proceedings.mlr.press/v108/ye20a.html.

Cun-Hui Zhang. Nearly unbiased variable selection under minimax concave penalty. *The Annals of Statistics*, 2010a.

Jian Zhang and Bernard Ghanem. Ista-net: Interpretable optimization-inspired deep network for image compressive sensing. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1828–1837. IEEE, 2018.

Tong Zhang. Analysis of multi-stage convex relaxation for sparse regularization. *Journal of Machine Learning Research*, 11(3), 2010b.

A Appendix

A.1 Convex clustering

Many familiar models arise as instances of eq. (2). For example, choosing the function $F(z) = \sum_{e\in E} \lambda_e \|z_e\|_2$ (or its square) for edge-wise weights $\lambda_e = \lambda\sqrt{w_e}$ yields a penalty $F(Ku) = \lambda \sum_e w_e \|u_i - u_j\|_2$ used in semi-supervised learning, network lasso, and convex clustering. Similarly, choosing $F(z) = \sum_{e\in E} \lambda_e |z_e|$ (or other ℓ_p norms) yields a graph total-variation penalty that appears in graph cuts Robertson et al. (2025b) and graph TV denoising.

In this paper, we focus on the particular case of convex clustering. Convex clustering Chi et al. (2025) is a continuous relaxation of clustering in which each data point $x_i \in \mathbb{R}^d$ is assigned a learnable representative (i.e., centroid), $u_i \in \mathbb{R}^d$. The objective balances two goals: each centroid should remain close to its original data point, while centroids connected by large graph weights should be encouraged to coincide via a nonsmooth term in the objective. The regularization parameter λ controls this tradeoff. For small λ, the centroids remain close to the input points; as λ increases, neighboring centroids fuse into common representatives, producing a hierarchy of coarser clusterings. This solution path is illustrated in Figure 4.

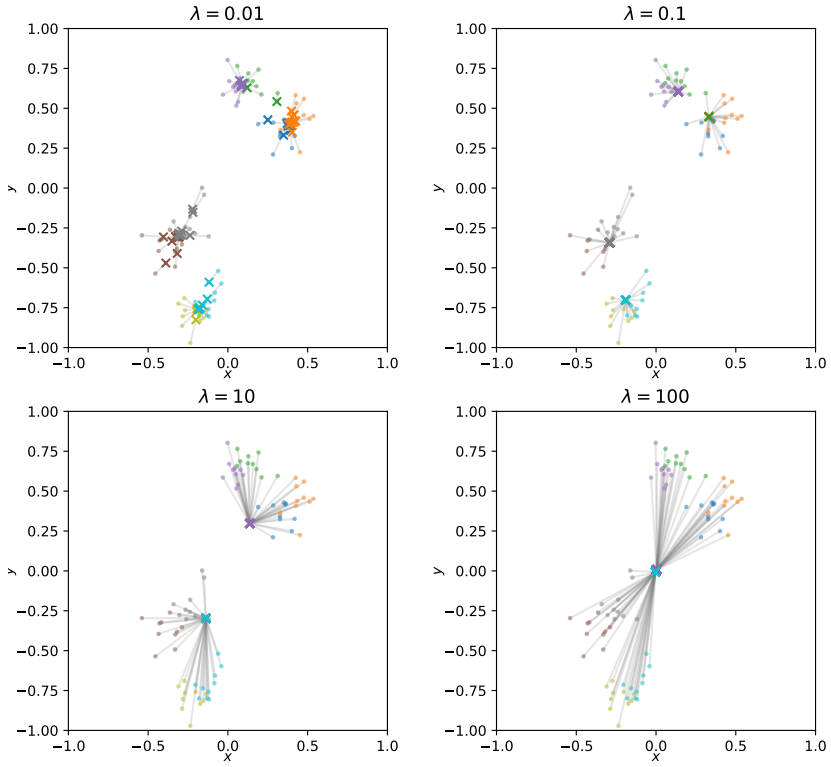

Figure 4: Solutions for convex clustering (eq. (5)) as λ increases on a k-NN graph for hierarchical Gaussians.

More concretely, classical convex clustering (and similarly Network Lasso) of points $x_1, \ldots, x_n \in \mathbb{R}^d$ can be written as a special case of eq. (2). We illustrate this in Table 2. Let the graph $\mathcal{G} = (V, E, w)$ have vertices $V = \{1, \ldots, n\}$, edge weights $w_e > 0$ for $e = (i, j) \in E$. Define

$$G(u) = \frac{1}{2} \sum_{i \in V} \|u_i - x_i\|_2^2, \quad F(z) = \sum_{e \in E} \lambda_e \|z_e\|_2,$$

where $\lambda_e := \lambda \sqrt{w_e}$. Then eq. (2) becomes

$$\min_{u \in \mathbb{R}^{|V| \times d}} \frac{1}{2} \sum_{i \in V} \|u_i - x_i\|_2^2 + \lambda \sum_{(i,j) \in E} w_{ij} \|u_i - u_j\|_2, \tag{12}$$

which is the standard convex clustering objective (with u_i the cluster centroids). Here, G can be thought of as the reconstruction error and F as the regularity penalty. Generally, eq. (5) is solved using variable substitution or splitting methods (Chi & Lange, 2015) or dual projected gradient methods (Pi et al., 2021). Recently, semismooth second-order methods have also been proposed (Sun et al., 2021). In the main text of our paper, we derive the PDHG updates for this problem and demonstrate how these updates are computationally amenable to local message passing.

A.2 Related Work

Optimization problems on graphs and primal–dual algorithms. Graph-structured optimization problems date back 20 years. The fused lasso is based on total-variation (TV) distances on a chain graph (Tibshirani et al., 2005), while the network lasso extends TV-style coupling to general graphs for prediction and clustering (Hallac et al., 2015). Closely related are the family of convex clustering problems on point clouds (Chi & Lange, 2015; Chen et al., 2014; 2015). These problems introduce continuous representatives, "centroids" for each sample and promote fusion along a similarity graph. Related applications include optimal transport on graphs, including linear, nonlinear, and constrained variants Peyré & Cuturi (2020)—and two-player matrix games on graphs Bapat & Tijs (1996). On the learning side, TV-based semi-supervised learning on graphs has been studied

both statistically and algorithmically (Jung et al., 2018; 2019), and TV-inspired GNN layers have been used as inductive biases for clustering and pooling (Hansen & Bianchi, 2023).

A complementary body of work studies nonconvex penalties as surrogates for ℓ_0-type objectives, including Smoothly Clipped Absolute Deviation (SCAD) (Fan & Li, 2001) and the Minimax Concave Penalty (MCP) (Zhang, 2010a). Algorithmically, coordinate-descent and proximal methods for these penalties have been studied in depth (Breheny & Huang, 2011), and theoretical analyses highlight settings where nonconvex regularization can achieve stronger support recovery guarantees (Loh & Wainwright, 2015; 2017). More recently, convex–nonconvex constructions have been proposed for graph TV to encourage edge preservation while maintaining global tractability (Heng et al., 2025).

Many of these objectives can be written as composite problems of the form $\min_u G(u) + F(Ku)$ with K a (weighted) incidence operator. As we will see, one can also consider an equivalent *saddle point* representation of the form $\min_u \max_y$. The Primal–Dual Hybrid Gradient (PDHG) method (Chambolle & Pock, 2011) is a standard first-order approach for such saddle-point formulations, and its practical efficiency often hinges on conditioning and step-size selection Applegate et al. (2021). Notably, combinatorial preconditioning schemes exploit graph structure to accelerate proximal algorithms for graph-TV-type problems (Möllenhoff et al., 2018; Ye et al., 2020).

Loop unrolling and neural algorithmic reasoning. A large literature on *learned unrolling* turns iterative optimization methods into trainable architectures by learning parameters such as step sizes, thresholds, or preconditioners (Gregor & LeCun, 2010; Zhang & Ghanem, 2018; Xie et al., 2019; yang et al., 2016; Chen et al., 2022; Sambharya et al., 2023). On graphs, GNNs have also been designed to mimic classical iterative algorithms and inherit stability/contraction properties (Yang et al., 2021), while neural algorithmic reasoning (NAR) aims to learn execution traces of algorithms and generalize to larger inputs (Veličković et al., 2020; Velickovic & Blundell, 2021; Veličković et al., 2022; Nerem et al., 2025). Beyond generic NAR, duality-based formulations have been proposed to jointly learn paired algorithms (e.g., max-flow/min-cut) (Numeroso et al., 2023), and a recent primal–dual NAR benchmark aligns primal/dual variables with a bipartite message-passing representation (He & Vitercik, 2025).

Most closely related to this work are learned primal–dual and PDHG-inspired networks. PDNAR (He & Vitercik, 2025) and PDHG-Net (Li et al., 2024) both derive unrolled primal–dual iterations for GNNs and show that sufficiently expressive parameterizations can recover classical PDHG updates. More concretely, PDNAR investigates simulation of discrete primal–dual approximation algorithms using a bipartite primal/dual representation and additional supervision obtained by running classical PDHG. PDHG-Net is closer algorithmically, but it is specialized to linear programs. In particular, PDHG-Net requires constraints specific to the LP formulations considered in (Li et al., 2024). The problems we consider do not naturally fit within that framework, so PDHG-Net cannot be used as a processor network in our architecture. In contrast, our framework targets generic saddle-point problems and learns directly from the primal–dual gap, without additional supervision. The mechanism is a graph-local message-passing realization of preconditioned PDHG designed for *generic nonlinear and nonsmooth* objectives of the form $G(u) + F(Ku)$. We also provide theoretical results that interpret the learned updates and size generalization through the lens of preconditioning and acceleration. In a broader optimization-on-graphs context, OptGNN demonstrates that message passing can mimic relaxation-based solvers for Max-CSPs (Yau et al., 2024), and surveys catalog many combinatorial-optimization uses of GNNs (Cappart et al., 2023). Our contribution is complementary: we target generic graph-structured composite convex objectives and give an explicit correspondence between standard message-passing layers and preconditioned PDHG as well as connect learned dynamics to classical acceleration ideas such as heavy-ball momentum (Polyak, 1964).

A.3 Other penalties and their proximal maps (NORMALIZE)

Because the edge penalty is separable,

$$F(z) = \sum_{e \in E} F_e(z_e),$$

both F^* and its proximal map are also separable across edges.

Table 2: Notation: convex clustering as an instance of eq. (2).

Object	Graph interpretation	Convex clustering interpretation
u_i	node variable	centroid for point x_i
$(Ku)_e$	weighted edge difference	discrepancy between neighboring centroids
p_e	edge dual variable	fusion forcing term
$K^\top p$	aggregate incident edge feedback	net force on centroid u_i from incident edges
$F(Ku)$	graph regularizer	penalty encouraging fused centroids
$F^*(p)$	dual constraint/penalty	feasible range of edge tensions
$G(u)$	node objective	fidelity / data reconstruction term

Table 3: Examples of separable convex and nonconvex penalties.

Name	**Penalty** $F_e(z)$
Convex penalties	
Group TV (ℓ_2)	$\lambda_e \|z\|_2$
Anisotropic TV (ℓ_1)	$\lambda_e \|z\|_1$
Max-norm TV (ℓ_∞)	$\lambda_e \|z\|_\infty$
Nonconvex penalties	
ℓ_0	$\lambda_e \mathbf{1}\{z \neq 0\}$
Clipped ℓ_1(Zhang, 2010b)	$\lambda_e \sum_{j=1}^d \min(\|z_j\|, t)$
SCAD (Fan & Li, 2001)	$\lambda_e \mathrm{SCAD}_a(\|z\|_2)$
MCP (Zhang, 2010a)	$\lambda_e \mathrm{MCP}_\gamma(\|z\|_2)$

Therefore, the normalization operator

$$\mathsf{NORMALIZE}(y) = \mathrm{prox}_{\sigma F^*}(y)$$

acts edgewise, and it suffices to compute the map for a single edge.

In this section we suppress the edge index and write F for a generic penalty on $\mathbb{R}^d$. See Table 3 for several examples of convex and non-convex penalties. We derive associated proximal below.

A.3.1 Convex penalties

For any norm $\|\cdot\|$ with dual norm $\|\cdot\|_*$, consider

$$F(z) = \lambda \|z\|.$$

Its convex conjugate is the indicator onto the dual norm ball Parikh & Boyd (2014):

$$F^*(p) = \mathbb{I}_{\{\|p\|_* \leq \lambda\}}(p)$$

Consequently,

$$\mathrm{prox}_{\sigma F^*}(y) = \arg\min_p \frac{1}{2}\|p - y\|_2^2 + \sigma \mathbb{I}_{\{\|p\|_* \leq \lambda\}}(p) = \mathrm{Proj}_{\{\|p\|_* \leq \lambda\}}(y),$$

which is simply the Euclidean projection onto the dual norm ball and is independent of σ.

ℓ_2 penalty Let $F(z) = \lambda \|z\|_2$. Then, $F^*(p) = \mathbb{I}_{\{\|p\|_2 \leq \lambda\}}(p)$, and

$$\mathsf{NORMALIZE}(y) = \mathrm{Proj}_{\{\|p\|_2 \leq \lambda\}}(y) = \begin{cases} y, & \|y\|_2 \leq \lambda, \\ \dfrac{\lambda}{\|y\|_2} y, & \|y\|_2 > \lambda. \end{cases}$$

ℓ_1 penalty Let $F(z) = \lambda \|z\|_1$. Then, $F^*(p) = \mathbb{I}_{\{\|p\|_\infty \leq \lambda\}}(p)$, and the NORMALIZE map is the coordinatewise clipping

$$[\mathsf{NORMALIZE}(y)]_j = \mathrm{Proj}_{\{\|p\|_\infty \leq \lambda\}}(y) = \mathrm{sign}(y_j) \min(|y_j|, \lambda), \;\; j = 1, \ldots, d.$$

ℓ_∞ penalty Let $F(z) = \lambda\|z\|_\infty$. Then, $F^*(p) = \mathbb{I}_{\{\|p\|_1 \le \lambda\}}(p)$, and

$$\textsf{NORMALIZE}(y) = \text{Proj}_{\{\|p\|_1 \le \lambda\}}(y).$$

A.3.2 NONCONVEX PENALTIES

The previous rules rely on convex duality. For nonconvex penalties, the standard convex-conjugate PDHG formulation no longer applies directly. Here, we define the NORMALIZE map using the direct proximal map

$$\textsf{NORMALIZE}^+(y) := \text{prox}_{\tau F}(y).$$

Below we describe two simple examples.

ℓ_0 penalty Consider

$$F(z) = \lambda \mathbf{1}\{z \neq 0\}.$$

The proximal problem is

$$\min_u \frac{1}{2\tau}\|u - y\|_2^2 + \lambda \mathbf{1}\{u \neq 0\}.$$

The solution set consists of two relevant candidates: $u = 0$ and $u = y$. Their objective values are

$$\frac{1}{2\tau}\|y\|_2^2 \text{ and } \lambda,$$

Therefore, the proximal map is the thresholding rule:

$$\text{prox}_{\tau\lambda\mathbf{1}\{\cdot\neq 0\}}(y) = \begin{cases} 0, & \|y\|_2 \le \sqrt{2\tau\lambda}, \\ y, & \|y\|_2 > \sqrt{2\tau\lambda}. \end{cases}$$

Clipped ℓ_1 Let

$$F(z) = \lambda \sum_{j=1}^{d} \min(|z_j|, t).$$

Since F is separable, the proximal map acts coordinatewise. The resulting scalar prox is a soft-thresholding rule up to the clipping level, and equals the identity beyond that level:

$$[\textsf{NORMALIZE}^+(y)]_j = \begin{cases} 0, & |y_j| \le \tau\lambda, \\ \text{sign}(y_j)\big(|y_j| - \tau\lambda\big), & \tau\lambda < |y_j| \le t + \dfrac{\tau\lambda}{2}, \\ y_j, & |y_j| > t + \dfrac{\tau\lambda}{2}. \end{cases}$$

MCP Let

$$F(z) = \text{MCP}_{\lambda,\gamma}(\|z\|_2),$$

where

$$\text{MCP}_{\lambda,\gamma}(r) = \begin{cases} \lambda r - \dfrac{r^2}{2\gamma}, & 0 \le r \le \gamma\lambda, \\ \dfrac{\gamma\lambda^2}{2}, & r > \gamma\lambda. \end{cases}$$

Because F is radial, $\text{prox}_{\tau F}(y)$ is collinear with y, so the problem reduces to a one-dimensional minimization in the radius. This gives the group firm-thresholding map

$$\textsf{NORMALIZE}^+(y) = \begin{cases} 0, & \|y\|_2 \le \tau\lambda, \\ \dfrac{\gamma}{\gamma - \tau}\left(1 - \dfrac{\tau\lambda}{\|y\|_2}\right) y, & \tau\lambda < \|y\|_2 \le \gamma\lambda, \qquad (\tau < \gamma). \\ y, & \|y\|_2 > \gamma\lambda, \end{cases}$$

A.4 Additional Experiments and Experimental Details

All models were constructed using PyTorch and PyTorch-Geometric. Each model was trained for 500 epochs on a training set consisting of 1000 graphs with 100 nodes each. All models were trained on 8 NVIDIA RTX A6000 GPUs. Additionally, all models were trained using the ADAM optimizer (Kingma & Ba, 2015) provided in PyTorch and trained using a learning rate of 0.001.

We note that since the GNN baselines output node states rather than an explicit primal–dual pair, we construct a surrogate dual variable for these baselines from the predicted primal variables by taking pairwise differences across edges in order to train against the primal–dual gap (i.e. the dual variable estimate is the projected variable $p_{ij} = h_i^{(\ell)} - h_j^{(\ell)}$ where $h^{(\ell)}$ are learned node features). These baselines test whether the gains of GRAPHPDHG can be explained by generic message passing alone. GIN, GraphSAGE, GCN, and GAT are node-state message-passing architectures with different aggregation mechanisms, while EdgeConv uses pairwise edge messages based on local feature differences.

A.4.1 Detailed dataset descriptions.

Here, we give more detailed exposition on how each graph dataset was constructed.

Hierarchical Gaussian Clusters Benchmark To empirically evaluate GRAPHPDHG on multi-scale geometric graphs, we introduce the Hierarchical Gaussian Benchmark.

The dataset $D = \{G_m, X_m\}_{m=1}^M$ consists of graphs and node features sampled from a recursive Gaussian hierarchy. Each node in the hierarchy at level ℓ has b_ℓ children. Starting from a root node $c_\emptyset = 0$, children are sampled recursively according to

$$c_{u,j} \sim \mathcal{N}(c_u, s_\ell^2 I_d),$$

where u indexes a node at level $\ell - 1$ and s_ℓ is the standard deviation associated with level ℓ. At the leaves of the hierarchy, node features are sampled as

$$x_i \sim \mathcal{N}(c_\alpha, s_L^2 I_d),$$

where $\alpha = (j_1, \ldots, j_L)$ denotes the path from the root to a leaf. Thus, each point has a hierarchical label corresponding to its full path in the latent tree, rather than a single flat cluster assignment. The total number of nodes is

$$|V| = n_{\text{leaf}} \prod_{\ell=1}^{L} b_\ell,$$

where n_{leaf} is the number of sampled points per leaf.

Given the sampled features, the graph topology is induced geometrically by constructing a k-nearest neighbor graph with Gaussian edge weights.

Image datasets For MNIST and Fashion-MNIST, we used variational autoencoders with 3 fully connected layers of sizes (784,400,20) and (784,400,30), respectively, followed by a symmetrically defined decoder. The autoencoder was trained for 100 epochs on each dataset. The autoencoder architecture, loss, and training are similar to Kingma & Welling (2013). For CIFAR-10, we utilize publicly available SimCLR embeddings Chen et al. (2020).

For each dataset, we construct subgraphs over the latent feature space. We used all available data to construct the graph, with $n = 70,000$ nodes for MNIST and Fashion-MNIST, and $n = 60,000$ nodes for CIFAR-10. The graph was constructed as a k-nearest neighbor graph with Gaussian edge weights given by

$$w_{ij} = \exp\left(-4||x_i - x_j||^2 / d_k(x_i)^2\right),$$

where x_i are the latent variables for image i, and $d_k(x_i)$ is the distance in the latent space between x_i and its kth nearest neighbor. We used $k = 10$ in all experiments and symmetrize W. In order to generate graphs for training, we subsample graphs of 100 nodes from the overall k-nearest neighbor graph.

A.4.2 ADDITIONAL EXPERIMENTS ON CONVEX CLUSTERING

In this section we provide additional results and illustrations of results presented in the main text. We provide further results on image datasets as well as an additional ablation on the edge memory component of GRAPHPDHG, a comparison of the empirical convergence rate versus the theoretical convergence rate, and an accounting of the computational costs of GRAPHPDHG.

Further results on image datasets In Table 4, we demonstrate the effectiveness of GRAPHPDHG for generating solutions with low primal–dual gap for each image dataset. Since all models were trained using the primal–dual gap, this measure directly measures whether the network produces a primal–dual pair that is close to satisfying the optimality conditions of the convex clustering problem. A lower initial gap is especially important for warm-starting subsequent solvers, since it indicates that the subsequent solver begins from a more informative primal–dual state rather than from a generic initialization.

Across all datasets and both λ levels, GRAPHPDHG achieves the lowest mean primal–dual gap. Both GAT and GCN exhibit a lack of explicit edge-state propagation. Compared with GAT, the improvement of GRAPHPDHG is particularly pronounced. Compared with GCN, GRAPHPDHG consistently improves the initialization quality, reducing the mean gap by approximately 42% and 22% on MNIST for $\lambda = 2.5$ and $\lambda = 10$, respectively, and by approximately 39% and 24% on Fashion-MNIST. On CIFAR-10, all methods have larger variance, reflecting the greater difficulty and heterogeneity of the instances, but GRAPHPDHG still gives the best average initialization for both regularization levels. These results support our claim that the PDHG-aligned node-edge architecture learns more useful primal–dual structure than node-only GNN baselines.

The ablations in Table 4 further clarify which parts of the architecture are responsible for the improvement. The "Unrolled PDHG" baseline tests whether simply embedding a fixed number of classical PDHG iterations is sufficient to produce a useful initialization. Although this baseline is algorithmically aligned with the target solver, it performs worse than GRAPHPDHG in all cases and is especially unstable on Fashion-MNIST and CIFAR-10. This suggests that fixed unrolling alone does not provide enough adaptivity across graph instances; the learned message-passing parameters in GRAPHPDHG are important for producing low-gap primal–dual states within a small number of layers.

The "GRAPHPDHG (- projection)" ablation removes the explicit projection/normalization step in the edge update. This ablation isolates the role of maintaining dual feasibility through the PDHG-inspired edge map. Removing this projection substantially degrades the initialization quality. On MNIST, the full model reduces the mean gap relative to this ablation by approximately 64% for $\lambda = 2.5$ and 37% for $\lambda = 10$. On Fashion-MNIST, the corresponding reductions are approximately 70% and 38%, while on CIFAR-10 they are approximately 12% and 10%. These results indicate that the projection is not merely a cosmetic architectural constraint: it helps the network produce dual variables that are better aligned with the feasible dual geometry of convex clustering.

These ablations indicate that the performance of GRAPHPDHG is not explained solely by using a node–edge architecture or by unrolling a known optimizer. The best results require both learned primal–dual message passing and the PDHG-inspired edge projection. This supports the central design principle of GRAPHPDHG: preserving the algorithmic structure of PDHG while allowing the local update maps to adapt to the graph and feature distribution.

Figure 5 compares the downstream effect of different initializations on SSNAL across MNIST, Fashion-MNIST, and CIFAR-10. The left column reports the mean first-hit iteration: the first SSNAL iteration at which the primal–dual gap falls below a prescribed threshold. Thus, lower curves indicate that the initialization allows SSNAL to reach a certified solution in fewer iterations. Across all three datasets, GRAPHPDHG yields the fastest or near-fastest convergence, with the largest gains appearing at stricter gap thresholds. The right column reports the primal–dual gap of the neural initialization before SSNAL refinement. GRAPHPDHG produces consistently lower initial gaps than the GAT and GCN benchmarks, supporting the claim that explicitly maintaining edge-dual states yields more informative primal–dual warm starts. CIFAR-10 is more variable, but the same qualitative trend remains: GRAPHPDHG gives a lower initial gap and reduces the number of SSNAL iterations needed for high-accuracy refinement.

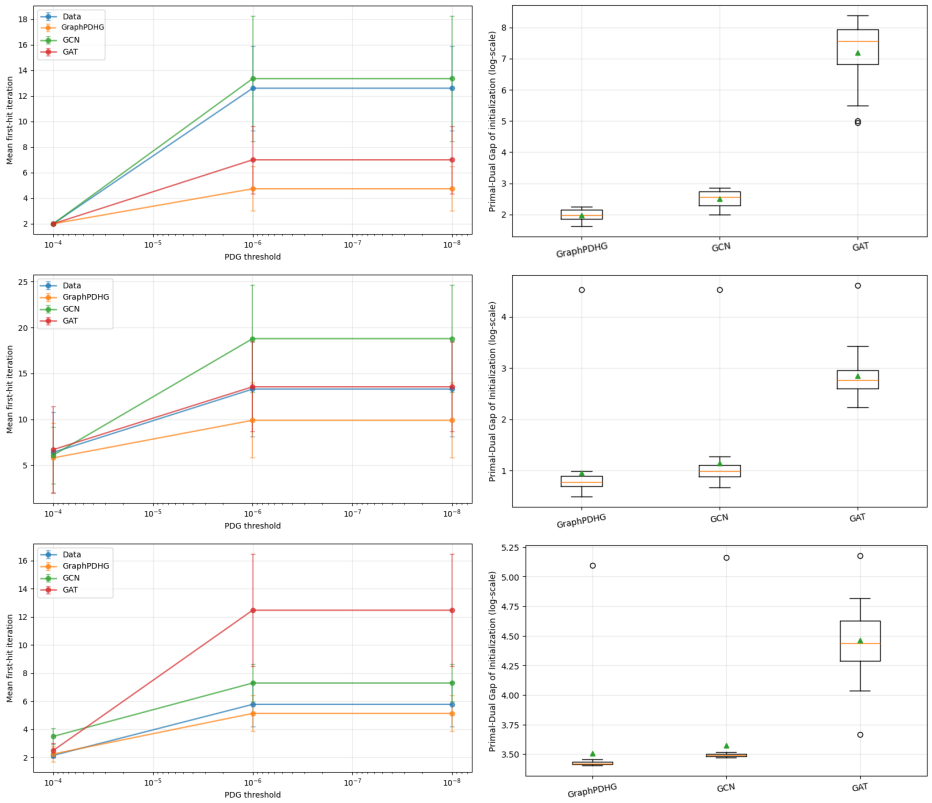

Figure 5: Comparing initializations. **Left Column:** Mean number of iterations of SSNAL to reach a primal–dual gap $\leq 10^{-6}$. **Right Column:** Primal-dual gap at initialization. **Top:** MNIST. **Middle:** Fashion-MNIST. **Bottom:** CIFAR-10.

Table 4: Primal–dual gap at initialization on image-derived convex clustering instances. Entries report mean ± standard deviation over runs. Lower is better, and the best result for each dataset and weight λ is bolded.

Dataset	Method	$\lambda = 2.5$	$\lambda = 10$
MNIST	GAT	1931.7 ± 1274.4	2167.9 ± 1544.4
	GIN	15.0 ± 4.5	98.0 ± 30.0
	GCN	12.6 ± 3.1	89.0 ± 24.9
	GraphSAGE	11.0 ± 2.8	82.0 ± 23.0
	EdgeConv	9.0 ± 2.0	74.0 ± 18.0
	Unrolled PDHG	64.3 ± 28.1	143.2 ± 59.3
	GRAPHPDHG (- projection)	20.4 ± 4.6	110.4 ± 41.3
	GRAPHPDHG	$\mathbf{7.3 \pm 1.3}$	$\mathbf{69.5 \pm 14.8}$
Fashion-MNIST	GAT	700.6 ± 541.8	764.3 ± 530.1
	GIN	12.5 ± 5.0	95.0 ± 31.0
	GCN	10.0 ± 3.5	84.6 ± 23.5
	GraphSAGE	8.8 ± 3.2	78.0 ± 22.0
	EdgeConv	7.3 ± 2.4	69.0 ± 18.0
	Unrolled PDHG	$8.7 \times 10^4 \pm 3.0 \times 10^3$	$9.4 \times 10^4 \pm 4.3 \times 10^3$
	GRAPHPDHG (- projection)	20.4 ± 19.1	104.3 ± 27.8
	GRAPHPDHG	$\mathbf{6.1 \pm 1.7}$	$\mathbf{64.2 \pm 14.9}$
CIFAR-10	GAT	27467.9 ± 3160.2	27693.4 ± 3256.8
	GIN	3300.0 ± 3200.0	3450.0 ± 3100.0
	GCN	3098.7 ± 3179.2	3244.3 ± 3017.8
	GraphSAGE	2950.0 ± 2600.0	3100.0 ± 2550.0
	EdgeConv	2750.0 ± 1200.0	2900.0 ± 1150.0
	Unrolled PDHG	$6.2 \times 10^6 \pm 1.4 \times 10^3$	$7.9 \times 10^6 \pm 1.7 \times 10^3$
	GRAPHPDHG (- projection)	2978.5 ± 74.7	3057.3 ± 83.1
	GRAPHPDHG	$\mathbf{2622.2 \pm 74.2}$	$\mathbf{2741.3 \pm 79.3}$

Table 5: Ablating the edge memory component of GRAPHPDHG. We measure the median first hit times for a target primal-dual gap of 0.001 after initializing SSNAL with GRAPHPDHG with edge memory vs. GRAPHPDHG without edge memory. Note that initializing SSNAL with GRAPHPDHG without the edge component, the median first hit is higher than if we had initialized SSNAL with GRAPHPDHG with an edge component.

	$\lambda = 0.001$	$\lambda = 0.01$	$\lambda = 0.10$	$\lambda = 1.0$
With edge memory	16.0	41.0	164.0	176.0
Without edge memory	18.0	53.0	161.0	182.0

Ablations on edge memory of GRAPHPDHG We performed additional ablations on the edge memory component of GRAPHPDHG for several values of λ on the hierarchical Gaussians dataset and record the median first hit for a threshhold of 10^{-6} below. By first hit, we mean the first iteration where each method reaches below the threshhold. Notice that without edge memory, GRAPHPDHG have a higher median first hit than with edge memory for lower $\lambda = 0.001, 0.01$. They have a comparable median hit rate as λ grows. However, this is because as λ grows, the convex clustering objective collapses into a single point (the global mean) and most baselines can find this degenerate solution. Nevertheless, this ablation points to the importance of the edge component of GRAPHPDHG (representing the dual variable) is essential for the performance of the method.

Empirical convergence rate versus accelerated convergence rate In Table 6, we compare the empirical primal-dual gap achieved by GRAPHPDHG on a dataset of expander graphs (with $\kappa = 2.53$), together with the accelerated and standard theoretical convergence rates for PDHG. Note that k in Table 6 indicates the iteration number for PDHG and the corresponding processor layer for GRAPHPDHG.

We observe that on the expander graphs, GRAPHPDHG reduces the median relative primal-dual gap below 10^{-2} using only two processor layers, while canonical PDHG first reaches this threshold

at 12 iterations. GRAPHPDHG is the top performing method through depth 14, with the two methods becoming comparable by depth 16. For the median full-Laplacian condition number $\kappa = 2.53$, the accelerated theoretical factor described in the main text is $q_\kappa = 0.228$, illustrating geometric decay compared with PDHG's normalized $1/k$ ergodic-rate shape. We believe that the gains begin to plateau at larger values of k because increasing k increases the depth of the message-passing network, making optimization more difficult. In practice, GNNs are often shallow (with relatively few message-passing layers), as deeper architectures can be susceptible to oversmoothing and oversquashing (see). Although these effects can be partially mitigated through residual connections (which are also included in our architecture), training and optimizing deep GNNs remains challenging. We believe that these depth-related optimization issues limit additional gains as k increases.

Table 6: Empirical gaps and theoretical convergence rates on expanders with $\kappa = 2.53$.

k	GraphPDHG empirical gap	PDHG empirical gap	Accelerated theory q_κ^k	PDHG theory $1/k$
1	1.90×10^{-2}	5.02×10^{-1}	2.28×10^{-1}	1.00
2	1.60×10^{-3}	3.26×10^{-1}	5.20×10^{-2}	5.00×10^{-1}
4	6.14×10^{-3}	1.41×10^{-1}	2.71×10^{-3}	2.50×10^{-1}
8	1.35×10^{-3}	3.06×10^{-2}	7.33×10^{-6}	1.25×10^{-1}
12	1.48×10^{-3}	7.52×10^{-3}	1.99×10^{-8}	8.33×10^{-2}
16	7.51×10^{-4}	6.89×10^{-4}	5.38×10^{-11}	6.25×10^{-2}

Computational cost of GRAPHPDHG Here, we provide a discussion of the computational cost of GRAPHPDHG. First, we note that our proposed GRAPHPDHG method is linear time per forward pass (as is the vanilla PDHG algorithm). All baselines are also linear time per forward pass so there is no significant difference in their theoretical complexity. Empirically, we record the following inference and training times for each method.

The training times recorded are for a training dataset of 1000 hierarchical Gaussian graphs with at most 100 nodes per graph and 15,000 training steps. The inference times recorded are for a single forward pass of each method on graphs with up to 100 nodes.

Table 7: Training time and inference times for each method on hierarchical Gaussian graphs with 100 nodes.

	GraphPDHG	GCN	GAT
Training time	1h 20m	1h 26m	1h 12m
Inference time (ms)	4.367	2.977	5.602

A.5 TOTAL VARIATION DENOISING ON GRAPHS

We evaluate GRAPHPDHG on graph-signal denoising to demonstrate its applicability beyond the original clustering experiments. We consider both convex total variation (TV) and nonconvex, nonsmooth capped-fusion regularization, and compare against classical solvers and parameter-matched neural baselines.

Signal generation. We generate connected, weighted PyGSP sensor graphs with $n = 244$ nodes on a randomly perturbed 12×12 grid in $[0, 1]^2$, using PyGSP's symmetrized k-nearest-neighbor construction with $k = 8$. To construct a clean signal, we sample four Voronoi centers uniformly in the unit square. Each region receives independent coefficients $a_r \sim \mathcal{N}(0, I_2)$ and $b_r \sim \mathcal{N}(0, 0.5^2)$. For node coordinates p_i, we define

$$s_i = a_{r(i)}^\top p_i + b_{r(i)}, \quad x_i^\star = \frac{s_i - \bar{s}}{\sqrt{n^{-1} \sum_j (s_j - \bar{s})^2}}, \quad y_i = x_i^\star + \varepsilon_i, \;\; \varepsilon_i \sim \mathcal{N}(0, 0.4^2). \tag{13}$$

Thus, each clean signal has zero mean and unit variance, and the observation noise has variance 0.16. We use 512 training, 128 validation, and 256 test graphs. The prediction task is to recover the

clean scalar signal from the noisy observations and graph. Clean signals serve as supervised training targets only; neither clean values nor region assignments are inputs at inference.

Reconstruction objectives. For each undirected edge $e = (i, j)$, define the weighted incidence operator $(Ku)_e = \sqrt{w_{ij}}(u_i - u_j)$. The two reconstruction objectives that we consider are

$$F_{\mathrm{TV}}(u) = \tfrac{1}{2}\|u - y\|_2^2 + \lambda_{\mathrm{TV}}\|Ku\|_1, \tag{14}$$

$$F_{\mathrm{cap}}(u) = \tfrac{1}{2}\|u - y\|_2^2 + \lambda_{\mathrm{cap}} \sum_{e \in E} \min\{|(Ku)_e|, c\}. \tag{15}$$

The convex TV problem has the saddle-point formulation

$$\min_u \max_{\|q\|_\infty \le \lambda_{\mathrm{TV}}} \tfrac{1}{2}\|u - y\|_2^2 + \langle Ku, q\rangle. \tag{16}$$

Capped fusion reduces the penalty on large jumps by saturating their contribution.

Methods We evaluate GRAPHPDHG against the GNN methods described in Section 4. Additionally, we distinguish between GRAPHPDHG-TV uses the TV operator throughout; GRAPHPDHG-MM starts with a TV-style update and then applies four iterations of the MM (majorize-minimization) algorithm recomputed from the current reconstruction. Note that GRAPHPDHG-MM approximately solves the capped-fusion proximal problem

$$\min_u \frac{1}{2}\|u - y\|_2^2 + \lambda \sum_{e \in E} \min\{|(Ku)_e|, c\}$$

At each iteration of the proximal map, we construct an upper bound by linearizing the concave function $s \mapsto \min\{s, c\}$ at the current edge magnitudes. Defining $m_e^{(t)} = \mathbf{1}\{|(Ku^{(t)})_e| < c\}$, the resulting subproblem is

$$\min_u \frac{1}{2}\|u - y\|_2^2 + \lambda \sum_{e \in E} m_e^{(t)} |(Ku)_e|.$$

Each iteration of MM alternatives between applying the TV-style proximal map and updating the mask m_e. We also compare against vanilla PDHG with the TV operator as well as vanilla PDHG with MM updates.

Results We measure both the normalized mean squared error (NMSE) between the reconstructed signal output (from each method) and the clean signal and the signal-to-noise ratio gain (SNR gain) which measures the difference between the input noisy graph signal and the output signals from each method. GRAPHPDHG-MM achieved the lowest NMSE at 3.33e-02, a 79% reduction from the noisy observation's 1.57e-01 and a 6.85 dB SNR improvement. GRAPHPDHG-TV and GAT were close at 3.37e-02 and 3.42e-02, respectively. GraphSAGE, EdgeConv, and GCN were also competitive, with NMSE values between 3.55e-02 and 3.68e-02. For CPU inference runtime, GraphSAGE was the fastest graph-based model at 2.03 ms/graph, followed by GCN at 2.89 ms/graph. GRAPHPDHG-TV and GRAPHPDHG-MM required approximately 4.12 ms/graph, while EdgeConv and GAT required 5.05 and 5.43 ms/graph, respectively. PDHG-MM was considerably slower at 10.73 ms/graph.

Table 8: Graph signal denoising. Comparison of reconstruction methods. Uncertainties are computed as half the reported interval width.

Method	NMSE	SNR gain (dB)	CPU (ms)
Noisy	$3.922 \times 10^{-2} \pm 6.75 \times 10^{-4}$	0.00 ± 0.00	0.000
Heat	$3.231 \times 10^{-2} \pm 6.85 \times 10^{-4}$	0.91 ± 0.08	0.816
PDHG-TV	$1.897 \times 10^{-2} \pm 5.65 \times 10^{-4}$	3.17 ± 0.15	3.184
PDHG-MM	$1.461 \times 10^{-2} \pm 9.10 \times 10^{-4}$	4.31 ± 0.28	11.373
GraphPDHG-TV	$1.075 \times 10^{-2} \pm 6.20 \times 10^{-4}$	5.63 ± 0.20	4.354
GraphPDHG-MM	$1.098 \times 10^{-2} \pm 5.80 \times 10^{-4}$	5.50 ± 0.18	4.367
MLP	$3.779 \times 10^{-2} \pm 5.55 \times 10^{-4}$	0.17 ± 0.025	0.670
GCN	$1.116 \times 10^{-2} \pm 6.90 \times 10^{-4}$	5.53 ± 0.16	2.977
GAT	$1.077 \times 10^{-2} \pm 5.25 \times 10^{-4}$	5.55 ± 0.155	5.602

B PROOFS AND ADDITIONAL RESULTS

Corollary B.1. *Suppose that G is μ-strongly convex:*

$$G(v) \geq G(u) + \langle g, v - u\rangle + \frac{\mu}{2}\|v-u\|_F^2 \qquad \forall u, v \in \mathbb{R}^{|V|\times d}, \text{ and } g \in \partial G(u).$$

Then $f(u) := G(u) + F(Ku)$ is also μ-strongly convex and admits a unique minimizer $u^\star$. Moreover, if $(u^\star, p^\star)$ is any saddle point of eq. (3), then for all $u \in \mathbb{R}^{|V|\times d}$ and $p \in \mathbb{R}^{|E|\times d}$,

$$\frac{\mu}{2}\|u - u^\star\|_F^2 \leq \mathcal{L}(u, p^\star) - \mathcal{L}(u^\star, p) \leq \mathit{Gap}(u, p). \tag{17}$$

From Theorem 2.2, we have that the PDHG iterates satisfy

$$\frac{\mu}{2}\|\bar{u}^{(N)} - u^\star\|_F^2 \leq \frac{1}{N}\left(\frac{s}{2}\|u^{(0)} - u^\star\|_S^2 + \frac{t}{2}\|p^{(0)} - p^\star\|_T^2\right), \tag{18}$$

Then,

$$\|\bar{u}^{(N)} - u^\star\|_F \leq \left(\frac{s\|u^{(0)} - u^\star\|_S^2 + t\|p^{(0)} - p^\star\|_T^2}{\mu N}\right)^{1/2}.$$

Proof of Theorem B.1. Let $(u^\star, p^\star)$ be any saddle point. By the saddle-point/KKT condition,

$$-K^\top p^\star \in \partial G(u^\star).$$

Applying strong convexity of G at $u^\star$ with subgradient $-K^\top p^\star$ gives

$$G(u) - G(u^\star) \geq \langle -K^\top p^\star,\ u - u^\star\rangle + \frac{\mu}{2}\|u - u^\star\|_F^2.$$

Therefore,

$$\begin{aligned}\mathcal{L}(u, p^\star) - \mathcal{L}(u^\star, p^\star) &= G(u) - G(u^\star) + \langle K(u - u^\star), p^\star\rangle \\ &\geq \frac{\mu}{2}\|u - u^\star\|_F^2.\end{aligned}$$

Since $p^\star$ maximizes $\mathcal{L}(u^\star, p)$, we also have

$$\mathcal{L}(u^\star, p^\star) \geq \mathcal{L}(u^\star, p) \qquad \forall p,$$

so

$$\frac{\mu}{2}\|u - u^\star\|_F^2 \leq \mathcal{L}(u, p^\star) - \mathcal{L}(u^\star, p).$$

For the upper bound, use

$$\mathcal{L}(u, p^\star) \leq \sup_q \mathcal{L}(u, q) = f(u), \qquad \mathcal{L}(u^\star, p) \geq \inf_v \mathcal{L}(v, p) = d(p),$$

which implies

$$\mathcal{L}(u, p^\star) - \mathcal{L}(u^\star, p) \leq f(u) - d(p) = \mathrm{Gap}(u, p).$$

Finally, apply Theorem 2.2 with $u = \bar{u}^{(N)}$ and $p = \bar{p}^{(N)}$. □

B.1 PROOF OF UNIVERSALITY FOR GRAPHPDHG

We first demonstrate that the message passing layer has the capacity to exactly recover one iteration of the PDHG algorithm, and subsequently inherits certain finite-layer convergence guarantees implied by Theorem 2.2. This notion of algorithmic alignment has been previously shown in related work He & Vitercik (2025) to be critical for generalization in NAR tasks.

Lemma B.2 (Representation)**.** *Let $\mathcal{A}_{PDHG}(\sigma, \tau)$ denote the operator defining one iteration of the preconditioned PDHG algorithm with fixed step sizes $\sigma, \tau > 0$. Assume the preconditioners have the form $S = \tau^{-1}I$ and $T = \sigma^{-1}I$. There exists a parameter configuration $\theta^* = \{A_1, A_2, B_1, B_2, B_3\}$ and a specific activation function for the message passing layers Φ_θ such that, if initialized with $h^{(0)} = u^{(0)}$ and $e^{(0)} = p^{(0)}$, the layer outputs exactly recover PDHG iterates:*

$$h^{(\ell)} = u^{(\ell)}, \quad e^{(\ell)} = p^{(\ell)} \quad \forall \ell \geq 1.$$

Proof. We proceed by constructively matching the update equations of the PDHG algorithm with the architecture of the proposed GNN layer. In particular, we observe that the GNN equations exactly match the PDHG updates under the following configuration θ^*:

Edge Update: Set $A_1 = I$, $A_2 = \sigma \cdot I$, and $\phi_{\text{edge}} = \text{Proj}_{B(\lambda)}(\cdot)$.

Node Update: Set $B_1 = \frac{1}{1+\tau} I$, $B_2 = \frac{\tau}{1+\tau} I$, $B_3 = -\frac{\tau}{1+\tau} I$.

Under this configuration, θ^*, the outputs $(h^{(\ell+1)}, e^{(\ell+1)}) = (u^{(\ell+1)}, p^{(\ell+1)})$. □

As a consequence of the above lemma, PDHG-derived message passing inherits the approximation and convergence guarantees of the primal–dual approximation algorithm it learns, as detailed in Theorem 3.1.

Proof of Theorem 3.1. The proof follows from the representation lemma, Lemma B.2 and standard results on the convergence rate of the PDHG algorithm.

If $\tau\sigma\|K\|^2 < 1$, the ergodic average of the iterates $(\bar{u}_L, \bar{p}_L) = \frac{1}{L}\sum_{k=1}^{L}(u_k, p_k)$ satisfies the following bound on the primal–dual gap:

$$\text{Gap}(\bar{u}_L, \bar{p}_L) \leq \frac{1}{L}\left(\frac{\|u^{(0)} - u^*\|^2}{2\tau} + \frac{\|p^{(0)} - p^*\|^2}{2\sigma}\right), \tag{19}$$

where (u^*, p^*) is a saddle point of the problem. Let $R^2 = \frac{\|u^{(0)}-u^*\|^2}{2\tau} + \frac{\|p^{(0)}-p^*\|^2}{2\sigma}$. The bound becomes

$$\text{Gap}_L \leq \frac{R^2}{L}.$$

To ensure the Gap is at most ϵ, we require $\frac{R^2}{L} \leq \epsilon \implies L \geq \frac{R^2}{\epsilon}$. Choose $\tau = \sigma = \frac{1}{1+\|K\|^2}$. Then $\tau\sigma\|K\|^2 < 1$, and R^2 scales with $\|K\|^2$. Thus, there exists a constant C depending only on initialization such that if

$$L = \frac{C \cdot \|K\|^2}{\epsilon}$$

the network achieves ϵ-optimality in the primal–dual gap. □

B.2 Analysis of linearized GraphPDHG

Proof of Theorem 3.4. We construct a specific parameterization. Consider the memoryless regime. Let $A_1 = 0$, and for simplicity let $B_1 = I_d$. Write down the error matrix using the terms in Remark 3.3:

$$\mathcal{E}^{(\ell+1)} = \mathcal{E}^{(\ell)} + K^\top(K\mathcal{E}^{(\ell)}A_2^\top)B_3^\top = \mathcal{E}^{(\ell)} + \Delta\mathcal{E}^{(\ell)}W$$

Where $W = A_2^\top B_3^\top \in \mathbb{R}^{d\times d}$. Choose $W = -\frac{1}{\lambda_k} I_d$ so

$$\mathcal{E}^{(\ell+1)} = \mathcal{E}^{(\ell)} - \Delta\mathcal{E}^{(\ell)}\frac{1}{\lambda_k} I_d$$

Left multiplying by $\psi_k^\top$ yields

$$\begin{aligned}\psi_k^\top \mathcal{E}^{(\ell+1)} &= \psi_k^\top \mathcal{E}^{(\ell)} - \psi_k^\top \Delta\mathcal{E}^{(\ell)}\frac{1}{\lambda_k} I_d \\ &= \psi_k^\top \mathcal{E}^{(\ell)} - \psi_k^\top \mathcal{E}^{(\ell)} = 0\end{aligned}$$

Since A_2 and B_3 are learnable $d \times d$ matrices, we can realize $W = -\frac{1}{\lambda_k} I_d$ by e.g. $A_2 = I_d$, $B_3 = -\frac{1}{\lambda_k} I_d$. □

Proof of Theorem 3.5. From the linearized update equations with $A_1 = \gamma I_d$ and $B_1 = I_d$, we have:

$$e^{(\ell+1)} = \gamma e^{(\ell)} + K\mathcal{E}^{(\ell)}A_2^\top \tag{20}$$

$$\mathcal{E}^{(\ell+1)} = \mathcal{E}^{(\ell)} + K^\top e^{(\ell+1)}B_3^\top \tag{21}$$

From eq. (21), we isolate the term $K^\top e^{(\ell+1)} B_3^\top = \mathcal{E}^{(\ell+1)} - \mathcal{E}^{(\ell)}$. Shifting the index back by one yields $K^\top e^{(\ell)} B_3^\top = \mathcal{E}^{(\ell)} - \mathcal{E}^{(\ell-1)}$. Multiply eq. (20) by $K^\top$ on the left and $B_3^\top$ on the right:

$$K^\top e^{(\ell+1)} B_3^\top = \gamma(K^\top e^{(\ell)} B_3^\top) + K^\top (K\mathcal{E}^{(\ell)} A_2^\top) B_3^\top.$$

Substituting the node difference expressions derived above:

$$(\mathcal{E}^{(\ell+1)} - \mathcal{E}^{(\ell)}) = \gamma(\mathcal{E}^{(\ell)} - \mathcal{E}^{(\ell-1)}) + \Delta\mathcal{E}^{(\ell)}(A_2^\top B_3^\top).$$

Define $M = -(A_2^\top B_3^\top)$. We recover the Polyak Heavy Ball update:

$$\mathcal{E}^{(\ell+1)} = \mathcal{E}^{(\ell)} + \gamma(\mathcal{E}^{(\ell)} - \mathcal{E}^{(\ell-1)}) - \Delta\mathcal{E}^{(\ell)} M.$$

□

Proof of Theorem 3.6. Consider the simplification $M = -A_2^\top B_3^\top = \eta I_d$. Then,

$$\mathcal{E}^{(\ell+1)} = \mathcal{E}^{(\ell)} + \gamma(\mathcal{E}^{(\ell)} - \mathcal{E}^{(\ell-1)}) - \eta\Delta\mathcal{E}^{(\ell)}.$$

Let $\Delta\psi_k = \lambda_k\psi_k$ such that $\|\psi_k\|_2 = 1$. Define the coefficient

$$c_k^{(\ell)} = \psi_k^\top \mathcal{E}^{(\ell)} \in \mathbb{R}^d$$

Left-multiply the recursion by $\psi_k^\top$:

$$\begin{aligned} c_k^{(\ell+1)} &= c_k^{(\ell)} + \gamma(c_k^{(\ell)} - c_k^{(\ell-1)}) - \eta\psi_k^\top \Delta\mathcal{E}^{(\ell)} \\ &= c_k^{(\ell)} + \gamma(c_k^{(\ell)} - c_k^{(\ell-1)}) - \eta\lambda_k\psi_k^\top \mathcal{E}^{(\ell)} \\ &= (1 + \gamma - \eta\lambda_k)c_k^{(\ell)} - \gamma c_k^{(\ell-1)} \end{aligned}$$

Equivalently, each coordinate of $c_k^{(\ell)}$ satisfies

$$c_{k,j}^{(\ell+1)} = (1 + \gamma - \eta\lambda_k)c_{k,j}^{(\ell)} - \gamma c_{k,j}^{(\ell-1)}$$

This recurrence has characteristic polynomial:

$$z^2 - (1 + \gamma - \eta\lambda_k)z + \gamma = 0. \tag{22}$$

Finally, Polyak (1964) shows that choosing

$$\eta^\star = \frac{4}{(\sqrt{\lambda_{\max}} + \sqrt{\lambda_{\min}})^2}, \quad \gamma^\star = \left(\frac{\sqrt{\lambda_{\max}} - \sqrt{\lambda_{\min}}}{\sqrt{\lambda_{\max}} + \sqrt{\lambda_{\min}}}\right)^2$$

yields a worst-case error which decays at the accelerated rate

$$\max_k \|c_k^{(\ell)}\|_2 \le C\left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^\ell$$

for some constant C. □

In the following Corollary, we tighten the analysis to consider the more general case where M is symmetric and positive definite.

Corollary B.3. *Assume the setting of Proposition 3.5, and suppose in addition that $M \in \mathbb{R}^{d\times d}$ is symmetric positive definite. Let $\Delta\psi_k = \lambda_k\psi_k$, and $\lambda_1 < \lambda_2 \le \cdots \le \lambda_{|V|}$, and let $Mq_j = \mu_j q_j$, and $\mu_j \ge 0, j = 1, \ldots, d$. Expand the error in the joint eigenbasis:*

$$\mathcal{E}^{(\ell)} = \sum_{k=1}^{|V|}\sum_{j=1}^{d} c_{k,j}^{(\ell)} \psi_k q_j^\top.$$

Then each joint mode satisfies the scalar recurrence

$$c_{k,j}^{(\ell+1)} = \big(1 + \gamma - \lambda_k\mu_j\big)c_{k,j}^{(\ell)} - \gamma c_{k,j}^{(\ell-1)}.$$

Define the joint condition number $\kappa_{\text{joint}} := \frac{\max_{k,j}\lambda_k\mu_j}{\min_{k,j}\lambda_k\mu_j}$. If $\gamma^\star := \left(\frac{\sqrt{\kappa_{\text{joint}}}-1}{\sqrt{\kappa_{\text{joint}}}+1}\right)^2$. Then there exists a constant $C > 0$, such that

$$\max_{k,j} |c_{k,j}^{(\ell)}| \le C\left(\frac{\sqrt{\kappa_{\text{joint}}} - 1}{\sqrt{\kappa_{\text{joint}}} + 1}\right)^\ell.$$

Proof. By Proposition 3.5,

$$\mathcal{E}^{(\ell+1)} = \mathcal{E}^{(\ell)} + \gamma\big(\mathcal{E}^{(\ell)} - \mathcal{E}^{(\ell-1)}\big) - \Delta\mathcal{E}^{(\ell)}M.$$

Expand

$$\mathcal{E}^{(\ell)} = \sum_{k=1}^{|V|}\sum_{j=1}^{d} c_{k,j}^{(\ell)}\psi_k q_j^\top,$$

where $\Delta\psi_k = \lambda_k\psi_k$ and $Mq_j = \mu_j q_j$. Then

$$\Delta(\psi_k q_j^\top)M = \lambda_k\mu_j\psi_k q_j^\top,$$

and the recurrence decouples over all of the modes (see the derivation in the previous corollary):

$$c_{k,j}^{(\ell+1)} = \big(1+\gamma-\lambda_k\mu_j\big)c_{k,j}^{(\ell)} - \gamma c_{k,j}^{(\ell-1)}.$$

Thus the (k,j)-th mode is governed by the heavy-ball recurrence. For the scalar heavy-ball method on a quadratic, the optimal worst-case asymptotic factor is

$$\rho^\star = \frac{\sqrt{\kappa_{\text{joint}}}-1}{\sqrt{\kappa_{\text{joint}}}+1}, \qquad \kappa_{\text{joint}} := \frac{\max_{k,j}\lambda_k\mu_j}{\min_{k,j}\lambda_k\mu_j}.$$

attained by

$$\gamma^\star = \left(\frac{\sqrt{\kappa_{\text{joint}}}-1}{\sqrt{\kappa_{\text{joint}}}+1}\right)^2.$$

Applying this uniformly yields $|c_{k,j}^{(\ell)}| \le C_{k,j}(\rho^\star)^\ell$. □

Proof of Theorem 3.7. From Proposition 3.5, there exists a parameter configuration such that the error dynamics follow the accelerated heavy-ball method. The asymptotic linear convergence rate ρ of this recurrence is given by:

$$\rho = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1} = 1 - \frac{2}{\sqrt{\kappa}+1}.$$

We need to show the following bound:

$$\|\mathcal{E}^{(\ell)}\|_2 \le O(\kappa)\|\mathcal{E}^{(0)}\|_2$$

Let $\epsilon > 0$. We want ℓ such that $\frac{\|\mathcal{E}^{(\ell)}\|}{\|\mathcal{E}^{(0)}\|} < \epsilon$. From the inequality shown above, we just need the minimum ℓ such that $\rho^\ell < \epsilon$. Since $\rho, \epsilon < 1$, $\ell\log(1/\rho) \ge \log(1/\epsilon)$ and $\ell \ge \log(1/\epsilon)\cdot\frac{1}{\log(1/\rho)}$.

Therefore, for any iteration ℓ such that $\ell \ge \log(1/\epsilon)\cdot\frac{1}{\log(1/\rho)}$, $\frac{\|\mathcal{E}^{(\ell)}\|}{\|\mathcal{E}^{(0)}\|} < \epsilon$. By Theorem B.2, there is a parameter configuration of GRAPHPDHG such that each layer implements one iteration of PDHG. Therefore, $\ell = \log(1/\epsilon)\cdot\frac{1}{\log(1/\rho)}$-layers of GRAPHPDHG can implement ℓ iterations of PDHG, so we get that the error at the ℓ-th layer of GNN is $\frac{\|\mathcal{E}^{(\ell)}\|}{\|\mathcal{E}^{(0)}\|} < \epsilon$. □