Archetypal Graph Generative Models: Explainable and Identifiable Communities via Anchor-Dominant Convex Hulls

Nikolaos Nakis[†] **Chrysoula Kosma**[‡] **Panagiotis Promponas**[†]
Michail Chatzianastasis[♣] **Giannis Nikolentzos**[♠]
[†]Yale Institute for Network Science, Yale University, USA
[‡]Université Paris Saclay, Université Paris Cité, ENS Paris Saclay, CNRS, SSA, INSERM, Centre Borelli, France
[♣]École Polytechnique, Paris, France
[♠]Department of Informatics and Telecommunications, University of Peloponnese, Greece

Abstract

Representation learning has been essential for graph machine learning tasks such as link prediction, community detection, and network visualization. Despite recent advances in achieving high performance on these downstream tasks, little progress has been made toward self-explainable models. Understanding the patterns behind predictions is equally important, motivating recent interest in explainable machine learning. In this paper, we present GraphHull, an explainable generative model that represents networks using two levels of convex hulls. At the global level, the vertices of a convex hull are treated as *archetypes*, each corresponding to a pure community in the network. At the local level, each community is refined by a *prototypical hull* whose vertices act as representative profiles, capturing community-specific variation. This two-level construction yields clear multi-scale explanations: a node's position relative to global archetypes and its local prototypes directly accounts for its edges. The geometry is well-behaved by design, while local hulls are kept disjoint by construction. To further encourage diversity and stability, we place principled priors, including determinantal point processes, and fit the model under MAP estimation with scalable subsampling. Experiments on real networks demonstrate the ability of GraphHull to recover multi-level community structure and to achieve competitive or superior performance in link prediction and community detection, while naturally providing interpretable predictions.

Proceedings of the 29[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

1 INTRODUCTION

In recent years, machine learning on graphs has attracted considerable attention. This is not surprising since data from different domains can be modeled as graphs. For instance, molecules (Gilmer et al., 2017), proteins (Gligorijević et al., 2021), and computer programs (Cheng et al., 2021) are commonly represented as graphs. Graph learning approaches mainly map elements of the graph into a low-dimensional vector space, while preserving the graph structure. A significant amount of research effort has been devoted to node embedding approaches, i. e., algorithms that map the nodes of a graph into a low-dimensional space (Cui et al., 2018). These methods typically use shallow neural network models to learn embeddings in an unsupervised manner. Those embeddings can then serve as input for various downstream tasks. Graph neural networks (GNNs) are another family of graph learning algorithms which are mainly applied to supervised learning problems. These models follow a message passing procedure, where each node updates its feature vector by aggregating the feature vectors of its neighbors (Wu et al., 2020).

Unsupervised learning and clustering methods play a central role in revealing latent structure and intrinsic organization in complex datasets. Among such approaches, *Archetypal Analysis* (AA) (Cutler and Breiman, 1994) provides a geometrically interpretable framework for representing data constrained to a K-dimensional polytope. In AA, each observation is expressed as a convex combination of extremal points, the *archetypes* which correspond to the vertices of the polytope and capture pure, representative profiles of

the data. Beyond its original formulation, AA has been studied extensively from both machine learning and geometric perspectives (Mørup and Hansen, 2012; Damle and Sun, 2015), highlighting connections to matrix factorization and convex geometry. Its geometric interpretation has also provided insight into trade-offs and Pareto-optimal structure in biological systems (Shoval et al., 2012). Over the years, AA has been successfully applied to diverse domains such as computer vision (Chen et al., 2014) and population genetics (Gimbernat-Mayol et al., 2022), while significant effort has been devoted to improving scalability and robustness (Eugster and Leisch, 2011; Bauckhage and Thurau, 2009). A comprehensive overview of methodological developments and applications is provided in the recent survey of Alcacer et al. (2025). More recently, AA has been generalized to relational settings (Nakis et al., 2025b, 2023a), enabling applications to network data.

One prominent application of graph learning algorithms is community detection (a.k.a., clustering), i. e., the problem of discovering clusters of nodes, with many edges joining nodes of the same cluster and comparatively few edges joining nodes of different clusters (Fortunato, 2010). In recent years, different graph learning algorithms have been proposed that can assign nodes to communities while maintaining high clustering quality (Li et al., 2018; Shchur and Günnemann, 2019; Chen et al., 2019b; Sun et al., 2020). Despite the success of these models in detecting communities in real-world networks, the lack of transparency still limits their application scope. Similar concerns have been raised in various domains, including healthcare and criminal justice (Rudin, 2019), and have led to the development of the field of explainable artificial intelligence (XAI) (Ribeiro et al., 2016; Ying et al., 2019; Gautam et al., 2022). XAI builds methods that justify a model's predictions, thus increasing transparency, trustworthiness and fairness. However, existing approaches generally focus on optimizing clustering quality rather than providing explanations for the learned communities. Moreover, current explainable graph learning methods mostly target node classification tasks, leaving the problem of interpretable community detection relatively underexplored. To the best of our knowledge, no existing approach provides a principled generative framework that jointly achieves accurate detection of communities and inherent interpretability.

In this paper, we introduce GraphHull, a novel generative framework for explainable graph representation learning. The method relies on archetypes (Cutler and Breiman, 1994), extreme profiles that summarize the diversity of structures found in a graph. Our contributions are: **i**) *Hierarchical geometric design* by proposing a two-level convex-hull architecture in which global archetypes define extreme community profiles and anchor-dominant local hulls introduce interpretable prototypes for community-specific variation; **ii**) *Identifiability by construction* in proving that local hulls are non-overlapping, which guarantees unique node reconstructions and transparent community assignments; **iii**) *Stable and scalable inference* by deriving a Lipschitz continuity result for the MAP objective, that enables provably safe optimization, and reducing likelihood evaluation from $\mathcal{O}(N^2)$ to $\mathcal{O}(|E|)$ via unbiased subsampling; **iv**) *Principled diversity* through determinantal point process priors on both global and local archetypes, that encourage non-degenerate and expressive latent geometry; and **v**) *Diverse empirical validation* via the competitive or superior performance of GraphHull on multiple real-world networks for link prediction and community detection, accompanied by interpretable multi-scale explanations. Overall, our geometric design enforces non-overlapping convex hulls anchored at global archetypes, yielding community embeddings that are simultaneously unique, interpretable, and diverse. *Code is available here:* `https://github.com/Nicknakis/GraphHull`

2 RELATED WORK

Graph learning for community detection. Community detection has been approached using both shallow embedding methods and deeper models such as GNNs. The standard pipeline embeds nodes into a low-dimensional vector space and then applies clustering algorithms like k-means. Unsupervised node representation learning methods include random walk–based approaches, matrix factorization, and autoencoders. Although shallow embeddings were once considered less effective, methods such as DeepWalk (Perozzi et al., 2014) and node2vec (Grover and Leskovec, 2016a) have recently been shown to achieve the optimal detectability limit under the stochastic block model (Kojaku et al., 2024). General-purpose embeddings often fail to capture community structure, motivating approaches that jointly preserve graph structure and optimize clustering objectives, e. g., modularity (Sun et al., 2020). Other works introduce community embeddings alongside node embeddings (Cavallari et al., 2017), or use nonnegative matrix factorization to directly encode community structure (Li et al., 2018). GNNs have been applied to community detection, e. g., with belief-propagation–inspired message passing (Chen et al., 2019b; Wang et al., 2025), Bernoulli–Poisson models for overlapping communities (Shchur and Günnemann, 2019), and node classification formulations (Wang et al., 2021). Autoencoder approaches embed nodes and then cluster them (Yang et al., 2016; Tian et al., 2014; Xie

et al., 2019), often with GNN encoders (Wang et al., 2017a; He et al., 2021), or unify embedding and clustering in a single framework (Wang et al., 2019; Chen et al., 2019a; Choong et al., 2018).

Explainable graph learning algorithms. Graph explainability methods fall into two categories: post-hoc approaches and self-explainable models. Post-hoc methods explain trained GNNs by identifying influential subgraphs (Ying et al., 2019; Yuan et al., 2020; Vu and Thai, 2020; Luo et al., 2020; Yuan et al., 2021; Lin et al., 2021; Henderson et al., 2021), but are sensitive to perturbations and thus less reliable (Li et al., 2024). Self-explainable models are interpretable by design, often via motifs (Lin et al., 2020; Serra and Niepert, 2024; Miao et al., 2022) or prototypes (Zhang et al., 2022; Ragno et al., 2022; Dai and Wang, 2021), with recent work formalizing their expressivity (Chen et al., 2024) and synthetic data for benchmarking (Agarwal et al., 2023). Our approach is related to prototype-based methods, but instead learns *archetypes*. In signed networks, archetypal embeddings have also been used for interpretable community structure (Nakis et al., 2025b,a).

3 PROPOSED METHOD

Preliminaries. Let $\mathcal{G} = (V, E)$ be a simple undirected graph with $N = |V|$ and adjacency matrix $\boldsymbol{Y} \in \{0,1\}^{N\times N}$, where $\boldsymbol{Y}_{ij} = \boldsymbol{Y}_{ji}$ and $\boldsymbol{Y}_{ii} = 0$. Capital bold letters such as $\mathbf{X}$ denote matrices, small bold letters such as $\mathbf{x}$ denote vectors, while non-bold letters such as x denote scalars. We next extend archetypal analysis to hierarchical nested structures by introducing a convex-hull formulation that captures latent structural organization in an inherently interpretable and generative manner, proposing GraphHull, illustrated in Figure 1. A high-level generative process of the model is provided in Algorithm 1 (the detailed generative process is provided in the supplementary material).

Global archetypes. Given a graph $\mathcal{G}$, we aim to embed nodes in a D-dimensional latent space that is constrained to a $(D-1)$-dimensional polytope. Each node's embedding is represented as a convex combination of the polytope's vertices, where each vertex corresponds to an *archetype* capturing a pure, extreme profile of the network (Nakis et al., 2023a, 2025b). Formally, we learn a matrix of archetypes

$$\boldsymbol{A} = [\boldsymbol{a}_1^\top; \ldots; \boldsymbol{a}_K^\top] \in \mathbb{R}^{K\times D} \quad \text{with } K \le D, \tag{1}$$

where each row defines a corner of the polytope. This geometric characterization couples communities with extreme archetypal profiles, yielding interpretable latent factors that explain network structure. Such an archetypal analysis of the network can successfully capture global characteristics, but it disregards potential structure within each archetype, motivating the need for a multi-scale extremal organization of the network.

Algorithm 1 Generative process

1: **Global archetypes:** Sample $\mathbf{A} \in \mathbb{R}^{K\times D}$ using the boxed SVD parameterization.
2: **Local hulls:** For each community $k = 1, \ldots, K$: construct prototypes $\mathbf{B}_k = \widetilde{\mathbf{W}}_k \mathbf{A}$ with anchor–dominant rows (truncated Dirichlet, ε). If $\varepsilon < \frac{1}{2}$, then $\mathrm{conv}(\mathbf{B}_k) \cap \mathrm{conv}(\mathbf{B}_\ell) = \varnothing$.
3: **Diversity priors:** Place DPP priors on rows of $\mathbf{A}$ and each $\mathbf{B}_k$ to encourage spread.
4: **Communities and nodes:** Draw community proportions $\boldsymbol{\pi}$, assignments $c_i \sim \mathrm{Cat}(\boldsymbol{\pi})$, barycentric weights $\boldsymbol{\omega}_i \sim \mathrm{Dir}(\alpha_\omega)$, and set $\boldsymbol{z}_i = \boldsymbol{\omega}_i^\top \mathbf{B}_{c_i}$. Draw degree bias $g_i \sim \mathcal{N}(0, \tau_g^2)$.
5: **Global scale:** Draw $s \sim \mathrm{HalfNormal}(\tau_s)$.
6: **Edges:** For each $i < j$, sample $Y_{ij} \sim \mathrm{Bern}(\sigma(s\, \boldsymbol{z}_i^\top \boldsymbol{z}_j + g_i + g_j))$.

A key design choice in our model is how to construct the global archetype matrix $\mathbf{A}$, which defines the vertices of the global convex hull in latent space. To ensure stability, we parameterize $\mathbf{A}$ in an SVD-like form,

$$\boldsymbol{A} \;=\; \boldsymbol{U}\,\mathrm{diag}(\boldsymbol{\sigma})\,\boldsymbol{V}^\top, \tag{2}$$

where $\boldsymbol{U} \in \mathbb{R}^{K\times K}$ and $\boldsymbol{V} \in \mathbb{R}^{D\times K}$ have orthonormal columns, and $\boldsymbol{\sigma} \in \mathbb{R}^K$ contains the singular values constrained to lie within fixed bounds $[\sigma_{\min}, \sigma_{\max}]$. This construction has three advantages: (i) it produces archetypes that span approximately orthogonal directions, reducing the risk of degeneracy; (ii) the bounded singular values control the overall scale of inner products, preventing either collapse or explosion of the latent embeddings; and (iii) the separation of orientation (via $\boldsymbol{U}, \boldsymbol{V}$) and scale (via σ) reduces redundant variability and stabilizes inference. Enforcing $\sigma_{\min} > 0$ guarantees that the rows of $\mathbf{A}$ are linearly independent, so that $\mathbf{A}$ always spans a well–conditioned latent basis.

Anchor–dominant local hulls. We posit K *global archetypes* collected in $\mathbf{A} \in \mathbb{R}^{K\times D}$, each serving as an extremal profile that defines a latent community in the network. To capture community–specific structure at a finer resolution, for each vertex (archetype) of the global hull $k \in \{1, \ldots, K\}$ we introduce a *local prototypical hull* $\mathbf{B}_k \in \mathbb{R}^{K\times D}$. Each node will be represented as a convex combination of the K rows of some $\mathbf{B}_k$, yielding interpretable community-aware prototypes. The idea parallels prototype-based modeling in interpretable image generation (Gautam et al., 2022; Kjærsgaard et al., 2024), but here it is adapted to relational data where both global and local structure must be inferred jointly.

At the global level, $\mathbf{A}$ contains extremal archetypes that define pure community profiles. At the local level, each hull $\mathbf{B}_k$ refines a community by introducing vertices that act as prototypes/local archetypes, i.e., representative subprofiles within the community. For enhanced

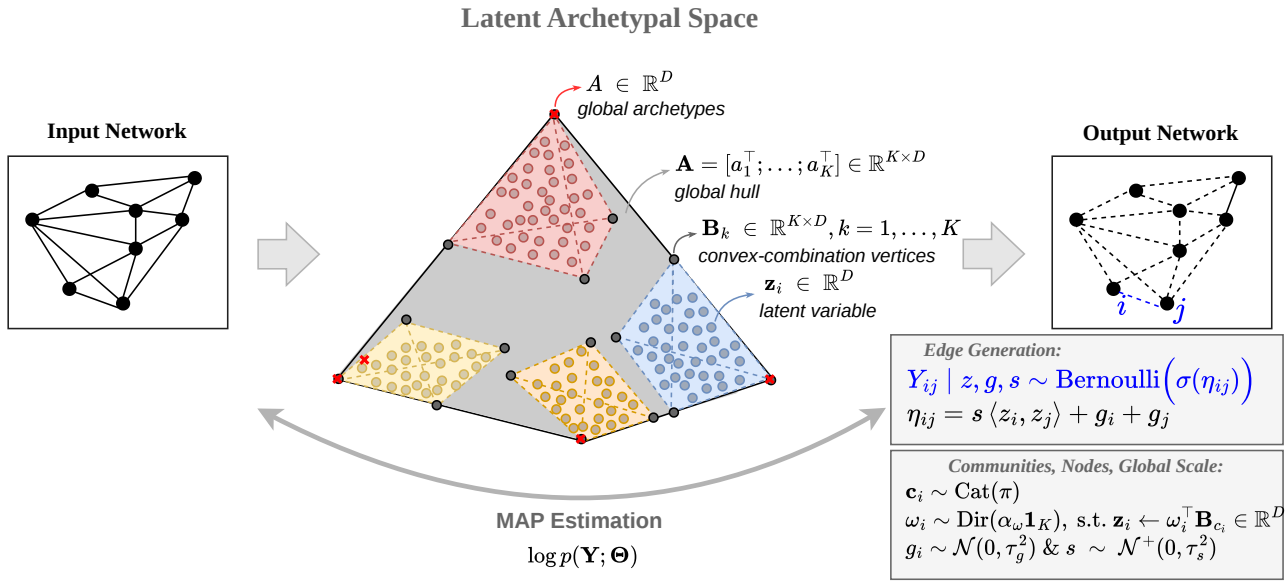


Figure 1: Visualization of the proposed archetypal graph generative model, so-called GraphHull.

identifiability, each local hull $\mathbf{B}_k$ is *anchored* to a distinct global archetype $\boldsymbol{a}_k^\top$. We write

$$\mathbf{B}_k \;=\; \tilde{\boldsymbol{W}}_k \boldsymbol{A} = \begin{bmatrix} \mathbf{W}_k \\ \boldsymbol{e}_k^\top \end{bmatrix} \mathbf{A} \;\in\; \mathbb{R}^{K \times D}, \qquad k = 1, \ldots, K, \tag{3}$$

where the total barycentric coordinates (including the anchor) are given by $\tilde{\boldsymbol{W}}_k$ so that $\boldsymbol{e}_k^\top \in \mathbb{R}^{1 \times K}$ is the k-th standard basis row (so the last row of $\mathbf{B}_k$ equals the anchor $\boldsymbol{a}_k^\top$), and $\mathbf{W}_k \in \mathbb{R}^{(K-1) \times K}$ is a *community-specific* row–simplex matrix whose rows are convex weights over the global archetypes:

$$(\mathbf{W}_k)_{\ell,:} \in \Delta_{K-1} \;:=\; \{\, \boldsymbol{w} \in \mathbb{R}^K : \; \boldsymbol{w} \geq 0, \; \mathbf{1}^\top \boldsymbol{w} = 1 \,\},$$
$$\ell = 1, \ldots, K-1.$$

Under a generative model, for selecting the anchor, one may regard this as sampling a global archetype without replacement and fixing it as a vertex of the local hull. The remaining $K-1$ vertices of $\mathbf{B}_k$ are constrained to be convex combinations of the global archetypes, ensuring that local extremes are explained hierarchically by the global structure, as prototypes. Thus, $\mathbf{B}_k$ has K rows (vertices in $\mathbb{R}^D$): one fixed anchor at the global archetype $\boldsymbol{a}_k^\top$ and $K-1$ prototypes with rows obtained as convex combinations of $\{\boldsymbol{a}_j^\top\}_{j=1}^K$. Importantly, $\mathbf{B}_1, \ldots, \mathbf{B}_K$ are *distinct* (each has its own coefficient matrix $\mathbf{W}_k$) and are anchored to different global archetypes.

Non–overlapping local hulls by construction. A central property of our design is *local hull identifiability*. Therefore, for each pair of communities $k \neq \ell$, their convex hulls should satisfy $\text{conv}(\mathbf{B}_k) \cap \text{conv}(\mathbf{B}_\ell) = \varnothing$. This disjointness ensures that every node embedding has a unique reconstruction: if two local hulls were to overlap, then any point in the intersection could be expressed as a convex combination of prototypes from either hull, breaking identifiability and interpretability of the representation. While one could attempt to discourage overlap by adding prior encouraging hull separation, these provide only a soft penalty and cannot guarantee non–overlap without careful tuning. Instead, we ensure disjointness *by construction* by defining the local hulls $\mathbf{B}_k$ through barycentric coordinates $\mathbf{W}_k$ sampled from a *truncated Dirichlet prior*. Specifically, each row $\boldsymbol{w}^\top$ is drawn as $\boldsymbol{w} \sim \text{Dir}(\alpha \mathbf{1}_K)$ conditioned on $w_k \geq 1 - \varepsilon$, where k denotes the anchor coordinate of hull k. This truncation forces every local prototype in the hull anchored at the k-th global archetype ($\boldsymbol{a}_k^\top$) to retain at least mass $1 - \varepsilon$ on its anchor. For $\varepsilon < \frac{1}{2}$ this ensures that anchor-dominant regions of different hulls are disjoint, so the induced convex hulls do not overlap cf. Lemma 3.1, (proof in supplementary).

Lemma 3.1 (Non-overlap of anchor-dominant local hulls)**.** *Let $\boldsymbol{A} \in \mathbb{R}^{K \times D}$ be affinely independent global archetypes. For each community $k \in \{1, \ldots, K\}$, let its local barycentric rows $\boldsymbol{w} \in \Delta_{K-1}$ be drawn from a truncated Dirichlet prior*

$$\boldsymbol{w} \sim \text{Dir}(\alpha \mathbf{1}_K) \;\; \textit{conditioned on } w_k \geq 1 - \varepsilon, \quad 0 < \varepsilon < \tfrac{1}{2}.$$

Let $\tilde{\boldsymbol{W}}_k$ stack K such rows, with the final row set to $\boldsymbol{e}_k^\top$, and define local vertices $\boldsymbol{B}_k = \tilde{\boldsymbol{W}}_k \boldsymbol{A}$. Then for any $k \neq \ell$,

$$\text{conv}(\boldsymbol{B}_k) \cap \text{conv}(\boldsymbol{B}_\ell) = \varnothing.$$

In practice we realize this truncated Dirichlet prior by parameterizing each row of $\boldsymbol{W}_k \in \mathbb{R}^{(K-1)\times K}$ directly. Let $\varepsilon \in (0, \frac{1}{2})$ be the anchor mass. For rows $r = 1, \ldots, K-1$ we set

$$(\mathbf{W}_k)_{r,:} = (1 - s_{k,r})\, \boldsymbol{e}_k^\top + s_{k,r}\, \tilde{\boldsymbol{q}}_{k,r}, \qquad (\mathbf{W}_k)_{r,:} \in \Delta_{K-1}, \;\; s_{k,r} = \varepsilon\, t_{k,r}, \;\; t_{k,r} \in (0,1), \tag{4}$$

where $\boldsymbol{e}_k$ is the k-th basis vector and $\tilde{\boldsymbol{q}}_{k,r} \in \Delta_{K-1}$ is a probability vector supported only on $\{1, \ldots, K\} \setminus \{k\}$ (i.e. zero mass on the anchor coordinate) where we can a place Dirichlet prior, $\tilde{\boldsymbol{q}}_k \sim \mathrm{Dir}(\alpha_q\, \mathbf{1}_K)$. The final row is set to $\boldsymbol{e}_k^\top$, yielding $\tilde{\boldsymbol{W}}_k \in \mathbb{R}^{K\times K}$, and the local vertices are obtained as $\mathbf{B}_k = \tilde{\boldsymbol{W}}_k \boldsymbol{A} \in \mathbb{R}^{K\times D}$. This anchor–dominant parameterization is equivalent to sampling from the truncated Dirichlet in Lemma 3.1, but is simpler to implement in practice and ensures the disjointness guarantee. Finally, the strength of anchor dominance in Eq. (4) is controlled by the shrink variable $s_{k,r} = \varepsilon\, t_{k,r}$, where $t_{k,r} \in (0,1)$ is given a Beta prior $t_{k,r} \sim \mathrm{Beta}(a, b)$.

Node representation. Having the global and local archetypal structure in place we can define the final node representation. For each node, we define a global community/archetype assignment to exactly one community (local hull). We encode this with the one–hot matrix, $\mathbf{M} \in \{0,1\}^{N\times K}$. Conceptually, each row $\boldsymbol{m}_i^\top$ corresponds to drawing the assignment c_i from a categorical distribution over the K global archetypes, so that $\boldsymbol{m}_i^\top$ is a one–hot vector indicating the unique hull to which node i belongs. If node i belongs to community c_i, then $\boldsymbol{m}_i^\top = \boldsymbol{e}_{c_i}^\top$ is the i-th row of $\mathbf{M}$. Within its community, the position of the node is given by barycentric coordinates $\boldsymbol{\omega}_i \in \Delta_{K-1}$ over the local archetypes $\mathbf{B}_{c_i} \in \mathbb{R}^{K\times D}$. Thus the embedding of node i is simply

$$\boldsymbol{z}_i \;=\; \boldsymbol{\omega}_i^\top\, \mathbf{B}_{c_i} \;\in\; \mathbb{R}^D. \tag{5}$$

Stacking all nodes yields the final representation matrix $\mathbf{Z} = [\boldsymbol{z}_1^\top; \ldots; \boldsymbol{z}_N^\top] \in \mathbb{R}^{N\times D}$ while $\Omega = [\boldsymbol{\omega}_1^\top; \ldots; \boldsymbol{\omega}_N^\top] \in \mathbb{R}^{N\times K}$ collects the barycentric weights. For the node–specific barycentric weights $\boldsymbol{\omega}_i$ over the K local archetypes of its assigned hull $\mathbf{B}_k$, we place a symmetric Dirichlet prior, $\boldsymbol{\omega}_i \sim \mathrm{Dir}(\alpha_\omega\, \mathbf{1}_K)$, with $\alpha_\omega = 1$, corresponding to the uniform distribution over the probability simplex. During training we employ the Gumbel–Softmax (GS) relaxation (Jang et al., 2017), over global assignment matrix $\mathbf{M} \in \{0,1\}^{N\times K}$ which provides a differentiable approximation to such categorical samples.

Edge likelihood. We focus on binary undirected graphs and model edges with a Bernoulli likelihood. The formulation can be extended to weighted graphs using a Poisson likelihood and to signed graphs using a Skellam likelihood. Let $g_i \in \mathbb{R}$ be a node-specific degree bias, $\boldsymbol{z}_i \in \mathbb{R}^D$ the latent representation of node i, and $s > 0$ a global scale parameter controlling geometric sharpness. For an unordered node pair (i, j) with $i < j$, the log-odds expression is

$$\eta_{ij} = s\, \langle \boldsymbol{z}_i, \boldsymbol{z}_j \rangle \;+\; g_i + g_j, \qquad Y_{ij} \mid \boldsymbol{Z}, \boldsymbol{g}, s \sim \mathrm{Bernoulli}\big(\sigma(\eta_{ij})\big). \tag{6}$$

The complete log-likelihood over the set of unordered pairs $\mathcal{D} = \{(i,j) : 1 \le i < j \le N\}$ is

$$\log p(\boldsymbol{Y} \mid \boldsymbol{Z}, \boldsymbol{g}, s) = \sum_{i<j} \Big[\, Y_{ij}\, \eta_{ij} - \log(1 + \exp \eta_{ij}) \,\Big]. \tag{7}$$

We place independent Gaussian priors on the node biases, $g_i \sim \mathcal{N}(0, \tau_{i,g}^2)$, while for the global scale parameter $s > 0$, we use a half-normal prior $s \sim \mathcal{N}^+(0, \tau_s^2)$.

Diversity of vertices within each local hull. An important characteristic of our model is the *diversity and expressiveness* of both the local hulls $\mathbf{B}_k$ and global hull $\boldsymbol{A}$. To capture rich community structure, each hull should span a large and well–conditioned volume, rather than collapsing onto a small subspace or producing co–linear vertices. To encourage this, we place a *Determinantal Point Process (DPP)* (Kulesza et al., 2012) prior in its L–ensemble formulation on the set of local/global archetypes. The L–ensemble DPP requires a positive–semidefinite kernel matrix.

For a convex hull, let $\boldsymbol{\Phi} \in \mathbb{R}^{K\times D}$ denote its K vertices and define the row–normalized matrix $\boldsymbol{\Psi}$, with rows $\boldsymbol{\psi}_k = \boldsymbol{\phi}_k / \|\boldsymbol{\phi}_k\|_2$, we then form the Gram matrix $\boldsymbol{L} = \boldsymbol{\Psi}\boldsymbol{\Psi}^\top$. The log–prior contribution is

$$\log p_{\mathrm{DPP}}(\boldsymbol{\Phi}) \;=\; \log\det(\boldsymbol{L}) \;-\; \log\det(\boldsymbol{I} + \boldsymbol{L}). \tag{8}$$

This determinantal point process prior acts as a repulsive force among both the local and global archetypes, discouraging degeneracy and collapse. Applied to the local hulls $\mathbf{B}_k$, it encourages the vertices to be diverse so that each hull spans a rich volume within its community. Applied to the global archetypes $\mathbf{A}$, it spreads the global basis vectors apart, ensuring that the communities themselves are well separated.

Joint distribution and MAP objective. Let Θ denote the GraphHull parameters, with deterministic constraints $\mathbf{B}_k = \tilde{\boldsymbol{W}}_k \mathbf{A}$ and $\boldsymbol{z}_i$ given by Eq. (5). Up to constants, the joint distribution factorizes as:

$$\log p(\boldsymbol{Y}, \boldsymbol{\Theta}) = \underbrace{\sum_{(i,j)\in\mathcal{D}} \Big[Y_{ij}\, \eta_{ij} - \mathrm{softplus}(\eta_{ij}) \Big]}_{\text{Bernoulli edge likelihood}} +$$

$$\underbrace{\sum_{i=1}^{N}(\alpha_{\boldsymbol{\omega}} - 1)\sum_{k=1}^{K} \log \omega_{ik}}_{\text{Dirichlet prior on } \omega_i} + \underbrace{\sum_{k=1}^{K}\sum_{r=1}^{K-1}(\alpha_q - 1)\sum_{j\neq k} \log(q_{k,r})_j}_{\text{non-anchor Dirichlet}}$$

$$+ \underbrace{\sum_{k=1}^{K}\sum_{r=1}^{K-1} \Big[(a-1)\log t_{k,r} + (b-1)\log(1 - t_{k,r}) \Big]}_{\text{anchor shrinkage (Beta)}}$$

$$+ \underbrace{\sum_{k=1}^{K} \Big[\log\det(\boldsymbol{L}_k) - \log\det(\boldsymbol{I} + \boldsymbol{L}_k) \Big]}_{\text{Local hull DPP prior}}$$

$$+ \underbrace{\Big[\log\det(\boldsymbol{L}) - \log\det(\boldsymbol{I} + \boldsymbol{L}) \Big]}_{\text{Global Hull DPP prior}} - \frac{1}{2\tau_g^2}\sum_{i=1}^{N} g_i^2 - \frac{s^2}{2\tau_s^2}\,. \tag{9}$$

Naively, evaluating the Bernoulli–logistic edge likelihood scales as $\mathcal{O}(N^2)$ due to all pairwise interactions. To achieve scalability, we exploit that the first term involves only observed edges and can be computed in $\mathcal{O}(|E|)$, while the log-partition term is estimated by uniform subsampling of non-edges, yielding an unbiased estimator. This reduces the per-iteration cost to $\mathcal{O}(|E|)$. The DPP prior adds a smaller overhead. For each local hull $\boldsymbol{B}_k \in \mathbb{R}^{K\times D}$ (and the global archetypes A), we compute a Gram matrix and two log-determinants, which require $\mathcal{O}(K^2D+K^3)$ operations. Since there are K local hulls plus one global term, the total DPP cost is $\mathcal{O}(K^3D + K^4)$. The remaining priors contribute lower-order costs. Thus, the overall per-iteration complexity of the method is $\mathcal{O}(|E|)+\mathcal{O}(K^3D+K^4)$, dominated by the linear-in-edges term for large graphs when $K \ll N$.

Optimization and curvature. Boxing the global archetype matrix not only improves identifiability but also controls curvature: the next theorem shows that the gradient with respect to the barycentric weights is globally Lipschitz, so fixed-step projected gradient is safe, as shown in Theorem 3.2 (proof in the supplementary). This helps explain the stable behavior we observe in practice.

Theorem 3.2 (Boxing ⇒ Lipschitz gradient in $\boldsymbol{\omega}$)**.** *Consider the negative Bernoulli–logistic loss*

$$\mathcal{L}(\boldsymbol{Z}, \boldsymbol{g}, s) = \sum_{i<j} \big[\log(1 + \exp \eta_{ij}) - Y_{ij}\eta_{ij} \big]. \tag{10}$$

Assume (i) the archetype matrix is boxed, $\|\mathbf{A}\|_2 \le \kappa_\star$; and (ii) each embedding $\mathbf{z}_i$ is a convex combination of rows of $\mathbf{A}$ (via local hull vertices), hence $\|\mathbf{z}_i\| \le \kappa_\star$. Let $\boldsymbol{\omega} = (\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_N)$ be the concatenated barycentric weights. Then the gradient $\nabla_{\boldsymbol{\omega}}\mathcal{L}$ is Lipschitz with constant

$$L_\omega \;\le\; \frac{1}{4}\left(s\,\kappa_\star^2\, \deg_{\max} \;+\; \tfrac{s^2}{2}\,\kappa_\star^4\, \deg_{\max} \right),$$

where $\deg_{\max} \le N - 1$ is the maximum degree. Consequently, projected gradient on the product of simplices with any stepsize $\eta \le 1/L_\omega$ is globally safe (monotone descent).

Remark (MAP). Adding the smooth priors in Eq. (9) preserves Lipschitz continuity: the total MAP gradient is Lipschitz with constant at most $L_{\omega,\mathrm{MAP}} \le L_\omega + L_{\omega,\mathrm{prior}}$, where $L_{\omega,\mathrm{prior}}$ is finite for Gaussian and log-determinantal terms. The Lipschitz bound depends on $s > 0$, the global scale parameter. In practice, s is regularized by a half-normal prior in the MAP objective keeping the Lipschitz constant finite.

Interpretability and Self-Explainability. Following the literature on self-explainable models (SEM) (Gautam et al., 2022), we characterize GraphHull through complementary notions of interpretability and explainability. The model is *interpretable* because its latent components, global archetypes and anchor-dominant local prototypes organized as nested convex hulls, correspond to extremal structural roles in the graph. It is *explainable* because each prediction admits an explicit generative decomposition: every node has a unique barycentric representation within a local hull, communities are anchored at global archetypes, and edge log-odds decompose into interpretable geometric interactions and degree effects. Thus, explanations arise directly from the model's latent geometry rather than post-hoc analysis. Moreover, GraphHull satisfies the core SEM predicates of *transparency*, *diversity*, and *trustworthiness* (Gautam et al., 2022). Transparency follows from the direct use of archetypes and prototypes in the generative mechanism. Diversity is enforced via affine independence, anchor-dominant disjoint hulls, and determinantal point process priors. Trustworthiness stems from faithful generative decompositions and the boxed-SVD parameterization, which yields stable optimization with controlled curvature. Finally, as a generative SEM, GraphHull enjoys *generative consistency*: the same interpretable latent factors that explain predictions also govern the probabilistic data-generation process. The hierarchical identifiability of prototypes through global archetypes further strengthens this alignment between geometry, explanation, and generation.

4 RESULTS

We extensively evaluate GraphHull against baseline graph representation learning methods on networks

Table 1: Statistics of networks. N: number of nodes, $|E|$: number of edges, K number of communities.

	LastFM	*Citeseer*	*Pol*	*Cora*	*Dblp*	*AstroPh*	*GrQc*	*HepTh*
N	7,624	3,327	18,500	2,708	27,199	17,903	5,242	8,638
$\|E\|$	55,612	9,104	61,200	5,278	66,832	197,031	14,496	24,827
K	14	6	2	7	—	—	—	—

of varying sizes and structures. All experiments were conducted on an 8 GB Apple M2 machine. For GraphHull, we optimize the MAP objective of Eq. (9) using Adam (Kingma and Ba, 2014) with learning rate in range $[0.01, 0.05]$. Unless otherwise stated, we set the anchor strength to $\varepsilon = 0.45$, box SVD bounds to $[\sigma_{\min} = 0.3, \sigma_{\max} = 1.5]$, and choose $K = D$ for simplicity. Detailed description of hyperparameters for all models are provided in the supplementary.

Datasets and Baselines. We evaluate on multiple undirected citation, collaboration, and social networks, each treated as unweighted (see Table 1 for details). Citation networks include *Cora* (machine learning publications, 7 classes) and *Citeseer* (computer science publications, 6 classes) (Sen et al., 2008). Collaboration networks include *DBLP* (co-authorship by research field) (Perozzi et al., 2017), *AstroPh* (astrophysics), *GrQc* (general relativity and quantum cosmology), and *HepTh* (high-energy physics) (Leskovec et al., 2007). We also use the social network *LastFM* (Asian users with 14 country labels) (Rozemberczki and Sarkar, 2020), and the socio-political retweet network *Pol* (labels denote political alignment) (Rossi and Ahmed, 2015). We compare against shallow embeddings, matrix factorization methods, mixed-membership models, and a GNN-based approach. Shallow embeddings include Node2Vec (Grover and Leskovec, 2016b) and Role2Vec (Ahmed et al., 2018). Factorization-based methods are NetMF (Qiu et al., 2018), GraRep (Cao et al., 2015), and RandNE (Zhang et al., 2018). Mixed-membership models include NNSED (Sun et al., 2017), MNMF (Wang et al., 2017b), and SymmNMF (Kuang et al., 2012). We also evaluate Dmon (Tsitsulin et al., 2023), which couples a GNN encoder with modularity maximization (for extended details see supplementary). We emphasize shallow embeddings, since they were recently shown to reach the detectability limit under the stochastic block model (Kojaku et al., 2024).

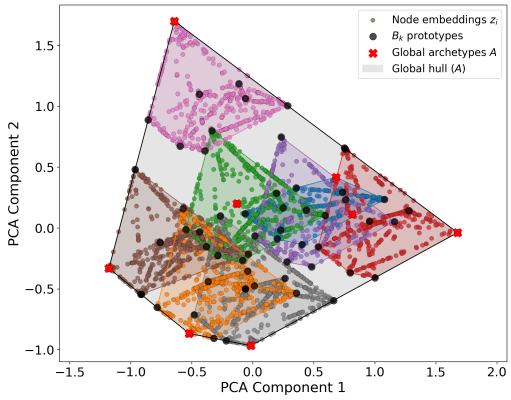
Link prediction. For link prediction, we follow the widely adopted evaluation protocol of Perozzi et al. (2014); Nakis et al. (2023b). Specifically, we randomly remove 50% of the edges while ensuring that the residual graph remains connected. The removed edges, together with an equal number of randomly sampled non-edges, form the positive and negative instances of the test set. The residual graph is then used to learn node embeddings. We evaluate performance on five benchmark networks, each over five runs and across multiple embedding dimensions ($D \in \{8, 16, 32, 64\}$). Table 2 reports the Area Under the Receiver Operating Characteristic Curve (AUC-ROC) (for Precision-Recall (PR-AUC) scores see supplementary). Across runs, the variance was consistently on the order of 10^{-3} and is omitted for readability. Following Grover and Leskovec (2016a), dyadic features are constructed using binary operators (average, Hadamard, weighted-L_1, weighted-L_2) and a logistic regression classifier with L_2 regularization is trained to make predictions. In contrast, predictions from GraphHull ($\epsilon = 0.49$) are obtained directly from the model: we use the log-odds η_{ij} of a test pair $\{i, j\}$ to compute link probabilities, with no additional classifier required. Importantly, GraphHull is tuned solely on training sets, and remains blind to the test set. By comparison, many baseline results are reported under test-aware tuning. Despite this advantage, GraphHull remains highly competitive with strong baselines across datasets and embedding dimensions, achieving comparable or superior performance in many settings while maintaining consistent link prediction accuracy.

Community detection. To assess community recovery, we use four networks with ground-truth labels. For membership-aware models, including GraphHull, we set the latent dimension equal to the number of true communities and compare inferred memberships with the ground truth. For GRL methods without memberships, we extract embeddings of the same dimensionality and apply k-means. We report Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI) averaging over five runs. All baselines are tuned for their main hyperparameters, while GraphHull uses the same settings across datasets. Results in Table 3 show GraphHull is consistently competitive with GRL baselines and outperforms membership-based methods, with the exception of *Cora*, where GRL with post-hoc clustering has an advantage. Nevertheless, GraphHull is the most consistent across datasets and metrics, outperforming membership-based methods, remaining competitive with GRL approaches.

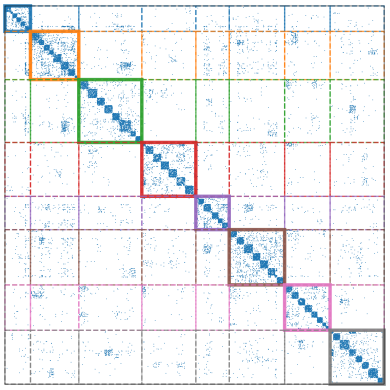
Network visualization. We illustrate how GraphHull captures global archetypal and local prototypical structures in the *HepTh* network. In Figure 2, panel (a) shows the PCA projection of the latent space inferred by GraphHull. Each local hull $\boldsymbol{B}_k$ is anchored in a distinct global archetype defined by the global hull $\boldsymbol{A}$. Any apparent overlap in panel (a) arises only from projecting to two dimensions; in the full D=8 space, all hulls are guaranteed to be non-overlapping. Panel (b) shows the adjacency matrix reordered by global hull assignments, and within each block by the maximum prototype membership $\arg\max \boldsymbol{\omega}_i$. This reveals

Table 2: AUC ROC scores for representation sizes of 8, 16, 32, and 64 averaged over five runs.

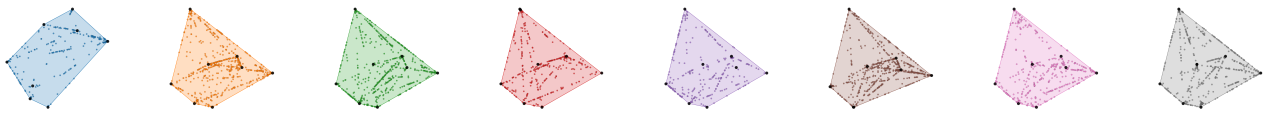
	AstroPh				*GrQc*				*HepTh*				*Cora*				*DBLP*			
Dimension (D)	8	16	32	64	8	16	32	64	8	16	32	64	8	16	32	64	8	16	32	64
NODE2VEC	.943	.954	.961	.962	**.928**	.932	.937	.936	**.879**	.882	.888	.892	.761	.760	.766	.777	.920	.923	.931	.941
ROLE2VEC	**.957**	**.969**	**.970**	**.965**	.927	**.936**	.934	.934	.897	**.907**	**.902**	.895	**.769**	.767	.759	.752	**.940**	**.952**	.943	.944
NETMF	.904	.928	.946	.955	.835	.882	.882	.883	.778	.797	.802	.793	.698	.675	.674	.654	.791	.817	.829	.842
GRAREP	.919	.946	.959	**.965**	.892	.906	.909	.894	.825	.845	.842	.829	.692	.704	.728	.717	.855	.877	.879	.867
RANDNE	.867	.888	.900	.907	.787	.826	.854	.877	.718	.770	.812	.839	.604	.641	.676	.701	.741	.795	.836	.865
NNSED	.863	.883	.891	.897	.799	.836	.861	.876	.741	.788	.820	.841	.620	.649	.689	.709	.755	.801	.834	.853
MNMF	.877	.916	.938	.954	.881	.905	.918	.918	.810	.844	.864	.875	.694	.718	.700	.680	.859	.898	.919	.931
SYMMNMF	.784	.818	.840	.862	.859	.873	.901	.905	.740	.770	.795	.826	.734	.717	.713	.705	.822	.848	.868	.888
DMON	.845	.852	.857	.861	.822	.845	.856	.866	.786	.799	.804	.807	.723	.744	.777	.788	.774	.785	.788	.791
GRAPHHULL	.945	.954	.960	**.965**	.923	.931	**.940**	**.945**	.874	.885	.895	**.904**	.753	**.790**	**.791**	**.803**	.930	.934	**.945**	**.948**



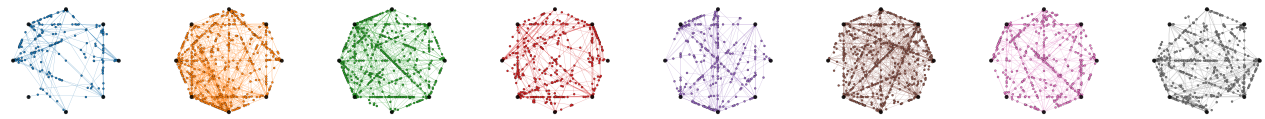
(a) GRAPHHULL PCA projection with $K = 8$ and $D = 8$.



(b) GRAPHHULL Reordered Adjacency matrix.



(c) PCA: $\boldsymbol{B}_1$ (d) PCA: $\boldsymbol{B}_2$ (e) PCA: $\boldsymbol{B}_3$ (f) PCA: $\boldsymbol{B}_4$ (g) PCA: $\boldsymbol{B}_5$ (h) PCA: $\boldsymbol{B}_6$ (i) PCA: $\boldsymbol{B}_7$ (j) PCA: $\boldsymbol{B}_8$



(k) CP: $\boldsymbol{B}_1$ (l) CP: $\boldsymbol{B}_2$ (m) CP: $\boldsymbol{B}_3$ (n) CP: $\boldsymbol{B}_4$ (o) CP: $\boldsymbol{B}_5$ (p) CP: $\boldsymbol{B}_6$ (q) CP: $\boldsymbol{B}_7$ (r) CP: $\boldsymbol{B}_8$

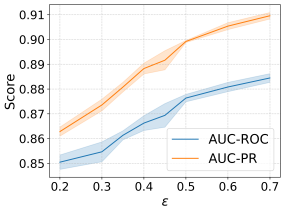
Figure 2: GRAPHHULL VISUALIZATIONS: (a) PCA projection of K=8 hulls in D=8 dimensions. (b) Reordered adjacency matrix: first by global hull assignments, then within blocks by $\arg\max \boldsymbol{\omega}_i$. (c)–(j) PCA projections of individual local hulls. (k)–(r) Prototype membership spaces for each $\boldsymbol{B}_k$, shown as circular plots (CP) with prototypes evenly spaced on the circle, with lines indicating links inside a hull (colors denote local hulls).

clear block structure at both the global and local levels. Panels (c)–(j) present disentangled PCA projections of each local hull $\boldsymbol{B}_k$. Finally, panels (k)–(r) show circular plots of the prototype membership spaces, where prototypes are positioned every $\frac{2\pi}{K}$ radians, illustrating node memberships $\boldsymbol{\omega}_i$ enriched with the links across nodes inside each block. Overall, GRAPHHULL extracts informative and robust network structures while combining the strengths of global archetypal characterization with prototypical structure inside communities.

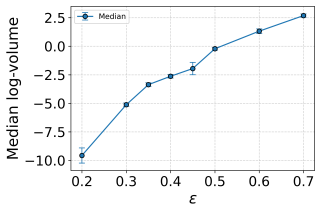
Effect of anchor strength ε. The parameter ε controls the maximum spread of each hull; for $\varepsilon < \frac{1}{2}$ it guarantees disjointness. Figure 3a shows link prediction performance as a function of ε. As ε increases, predictive performance steadily improves in both AUC-ROC and AUC-PR. This trend continues even in the overlapping regime ($\varepsilon > \frac{1}{2}$), where allowing overlaps better explains mixed-memberships. Notably, at $\varepsilon \approx 0.45$ we achieve high predictive accuracy and remain within the identifiable regime. We investigate hull geometry as ε varies. Figure 3b reports the effective log-volume across runs: volumes increase monotonically with ε, showing that hulls naturally expand as anchor dominance re-

Table 3: Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI) scores.

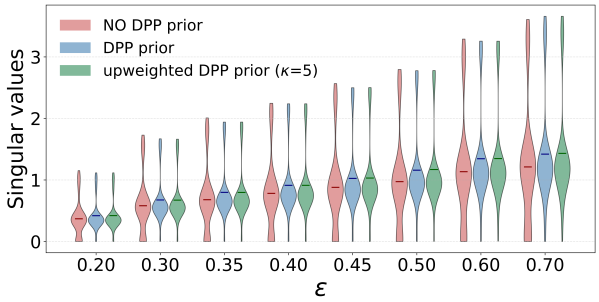
	Cora		*Citeseer*		*LastFM*		*Pol*	
Metric	NMI	ARI	NMI	ARI	NMI	ARI	NMI	ARI
NODE2VEC	**.460** ± .010	**.397** ± .024	.223 ± .010	.211 ± .025	.591 ± .005	.436 ± .009	.727 ± .007	.791 ± .008
ROLE2VEC	.486 ± .001	.417 ± .010	.299 ± .016	.279 ± .015	.596 ± .001	.455 ± .003	.712 ± .003	.772 ± .003
NETMF	.463 ± .006	.349 ± .006	.223 ± .010	.145 ± .016	.605 ± .001	**.522** ± .001	.145 ± .001	.199 ± .001
GRAREP	.292 ± .001	.192 ± .001	.320 ± .001	.320 ± .001	.394 ± .001	.259 ± .001	.447 ± .001	.492 ± .001
RANDNE	.020 ± .004	.010 ± .004	.015 ± .002	.010 ± .003	.063 ± .001	.018 ± .001	.003 ± .001	.003 ± .001
NNSED	.310 ± .031	.229 ± .027	.244 ± .011	.201 ± .016	.290 ± .006	.143 ± .011	.085 ± .043	.136 ± .056
MNMF	.294 ± .017	.226 ± .018	.144 ± .031	.125 ± .030	.473 ± .006	.307 ± .012	.599 ± .024	.624 ± .030
SYMMNMF	.311 ± .001	.204 ± .001	.310 ± .006	.289 ± .003	.448 ± .020	.329 ± .043	.117 ± .006	.153 ± .004
DMON	.350 ± .024	.299 ± .042	.271 ± 0.018	.265 ± 0.027	.518 ± .025	.397 ± .049	.714 ± .001	.769 ± .001
GRAPHHULL	.422 ± .004	.333 ± .008	**.347** ± .007	**.372** ± .009	**.616** ± .001	.514 ± .003	**.757** ± .001	**.820** ± .001



(a) Predictive performance as a function of ε.



(b) Effective log-volume of hulls as a function of ε.



(c) Violin plots of local-hull singular values as a function of the anchor strength ε under three priors: none (no DPP), standard DPP, and upweighted DPP ($\kappa = 5$).

Figure 3: EFFECT OF ε: Results on the *HepTh* dataset.

laxes. Although the DPP prior discourages coincident prototypes, it does not enforce full affine independence per hull. In practice, some communities form hulls of *reduced effective rank*, where a prototype is nearly an affine combination of others. This reflects benign over-parameterization: the data does not require all directions. Such rank deficiencies are harmless, the redundant vertex can be trivially removed without changing the hull, providing an interpretable estimate of a community's intrinsic dimension. Concretely, replacing $\boldsymbol{L}_k$ with $\kappa\boldsymbol{L}_k$ for $\kappa > 0$ scales the eigenvalues of the kernel, yielding the prior $\log\det(\kappa\boldsymbol{L}_k) - \log\det(\boldsymbol{I} + \kappa\boldsymbol{L}_k)$. Larger κ strengthens the repulsion between prototypes, discouraging redundant vertices without altering the feasible set of hulls. This effect is evident in Figure 3c: without a DPP prior, the singular values of local hulls $\boldsymbol{B}_k$ exhibit heavy tails near zero, producing unstable hulls with rank deficiency. Including a DPP prior almost entirely eliminates this issue, while upweighting with $\kappa = 5$ further removes rank deficiency.

Prototype interpretability. Examining the learned prototypes reveals clear structural roles. Across datasets, anchors generally correspond to dense cores, while non-anchor prototypes capture sparser or peripheral structures, highlighting diversity within each community. For example, in the *HepTh* network anchors exhibit systematically higher clustering coefficients than non-anchor prototypes, whereas peripheral prototypes often attract large numbers of "on-vertex" exemplars despite low clustering values. These findings confirm that the anchor–prototype hierarchy yields interpretable profiles of structural organization (see supplementary for detailed quantitative analysis).

5 CONCLUSION & LIMITATIONS

We introduced GRAPHHULL, a principled generative framework for explainable graph representation learning. By combining global archetypes with anchor–dominant local convex hulls, our approach yields embeddings that are identifiable, interpretable, and diverse by design. The geometry enforces disjointness and stability, while determinantal point process priors promote non-degenerate structure. Across multiple networks, GraphHull achieves competitive or superior performance on link prediction and community detection, while naturally providing multi-scale explanations of communities and prototypes. Like most generative models, optimization is non-convex and depends on initialization, but in practice we find GRAPHHULL to be stable under deterministic initializations based on the normalized Laplacian. Beyond graph analysis, the proposed hierarchical archetypal design is broadly applicable to other domains where transparent latent geometry is needed. We hope this work encourages further research into generative, geometry-aware models for interpretable machine learning.

ACKNOWLEDGEMENTS

We gratefully acknowledge the reviewers for their constructive feedback and insightful comments. N.N. is supported by the NOMIS foundation. C. K. is supported by the IdAML Chair hosted at ENS Paris-Saclay, Université Paris-Saclay.

References

C. Agarwal, O. Queen, H. Lakkaraju, and M. Zitnik. Evaluating explainability for graph neural networks. *Scientific Data*, 10(1):144, 2023.

N. K. Ahmed, R. Rossi, J. B. Lee, T. L. Willke, R. Zhou, X. Kong, and H. Eldardiry. Learning role-based graph embeddings. *arXiv preprint arXiv:1802.02896*, 2018.

A. Alcacer, I. Epifanio, S. Mair, and M. Mørup. A survey on archetypal analysis, 2025. URL `https://arxiv.org/abs/2504.12392`.

C. Bauckhage and C. Thurau. Making archetypal analysis practical. pages 272–281, 09 2009. ISBN 978-3-642-03797-9. doi: 10.1007/978-3-642-03798-6_28.

S. Cao, W. Lu, and Q. Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 891–900, 2015.

S. Cavallari, V. W. Zheng, H. Cai, K. C.-C. Chang, and E. Cambria. Learning community embedding with community detection and node embedding on graphs. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 377–386, 2017.

Y. Chen, J. Mairal, and Z. Harchaoui. Fast and robust archetypal analysis for representation learning. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1478–1485, 2014. doi: 10.1109/CVPR.2014.192.

Y. Chen, Y. Bian, B. Han, and J. Cheng. How interpretable are interpretable graph neural networks? In *Proceedings of the 41st International Conference on Machine Learning*, pages 6413–6456, 2024.

Z. Chen, C. Chen, Z. Zhang, Z. Zheng, and Q. Zou. Variational graph embedding and clustering with laplacian eigenmaps. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 2144–2150, 2019a.

Z. Chen, L. Li, and J. Bruna. Supervised community detection with line graph neural networks. In *7th International Conference on Learning Representations*, 2019b.

X. Cheng, H. Wang, J. Hua, G. Xu, and Y. Sui. Deepwukong: Statically detecting software vulnerabilities using deep graph neural network. *ACM Transactions on Software Engineering and Methodology*, 30 (3):1–33, 2021.

J. J. Choong, X. Liu, and T. Murata. Learning community structure with variational autoencoder. In *Proceedings of the 2018 IEEE International Conference on Data Mining*, pages 69–78, 2018.

P. Cui, X. Wang, J. Pei, and W. Zhu. A survey on network embedding. *IEEE Transactions on Knowledge and Data Engineering*, 31(5):833–852, 2018.

A. Cutler and L. Breiman. Archetypal analysis. *Technometrics*, 36(4):338–347, 1994.

E. Dai and S. Wang. Towards self-explainable graph neural network. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*, pages 302–311, 2021.

A. Damle and Y. Sun. A geometric approach to archetypal analysis and non-negative matrix factorization, 2015. URL `https://arxiv.org/abs/1405.4275`.

M. J. Eugster and F. Leisch. Weighted and robust archetypal analysis. *Computational Statistics and Data Analysis*, 55(3):1215–1225, 2011. ISSN 0167-9473. doi: https://doi.org/10.1016/j.csda.2010.10.017.

S. Fortunato. Community detection in graphs. *Physics reports*, 486(3-5):75–174, 2010.

S. Gautam, A. Boubekki, S. Hansen, S. A. Salahuddin, R. Jenssen, M. M. Höhne, and M. Kampffmeyer. Protovae: a trustworthy self-explainable prototypical variational model. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pages 17940–17952, 2022.

J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1263–1272, 2017.

J. Gimbernat-Mayol, A. Dominguez Mantes, C. D. Bustamante, D. Mas Montserrat, and A. G. Ioannidis. Archetypal analysis for population genetics. *PLoS Computational Biology*, 18(8):e1010301, 2022.

V. Gligorijević, P. D. Renfrew, T. Kosciolek, J. K. Leman, D. Berenberg, T. Vatanen, C. Chandler, B. C. Taylor, I. M. Fisk, H. Vlamakis, et al. Structure-based protein function prediction using graph convolutional networks. *Nature communications*, 12(1): 3168, 2021.

A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM ACM SIGKDD International Conference on*

Knowledge Discovery and Data Mining, pages 855–864, 2016a.

A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 855–864, 2016b.

D. He, Y. Song, D. Jin, Z. Feng, B. Zhang, Z. Yu, and W. Zhang. Community-centric graph convolutional network for unsupervised community detection. In *Proceedings of the 29th International Conference on International Joint Conferences on Artificial Intelligence*, pages 3515–3521, 2021.

R. Henderson, D.-A. Clevert, and F. Montanari. Improving molecular graph neural network explainability with orthonormalization and induced sparsity. In *Proceedings of the 38th International Conference on Machine Learning*, pages 4203–4213, 2021.

E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations*, 2017.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

R. Kjærsgaard, A. Boubekki, and L. Clemmensen. Pantypes: Diverse representatives for self-explainable models. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence*, pages 13230–13237, 2024.

S. Kojaku, F. Radicchi, Y.-Y. Ahn, and S. Fortunato. Network community detection via neural embeddings. *Nature Communications*, 15(1):9446, 2024.

D. Kuang, C. Ding, and H. Park. Symmetric nonnegative matrix factorization for graph clustering. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, pages 106–117, 2012.

A. Kulesza, B. Taskar, et al. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286, 2012.

J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):2–es, 2007.

Y. Li, C. Sha, X. Huang, and Y. Zhang. Community detection in attributed graphs: an embedding approach. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 338–345, 2018.

Z. Li, S. Geisler, Y. Wang, S. Günnemann, and M. van Leeuwen. Explainable graph neural networks under fire. *arXiv preprint arXiv:2406.06417*, 2024.

C. Lin, G. J. Sun, K. C. Bulusu, J. R. Dry, and M. Hernandez. Graph neural networks including sparse interpretability. *arXiv preprint arXiv:2007.00119*, 2020.

W. Lin, H. Lan, and B. Li. Generative causal explanations for graph neural networks. In *Proceedings of the 38th International Conference on Machine Learning*, pages 6666–6679, 2021.

D. Luo, W. Cheng, D. Xu, W. Yu, B. Zong, H. Chen, and X. Zhang. Parameterized explainer for graph neural network. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pages 19620–19631, 2020.

S. Miao, M. Liu, and P. Li. Interpretable and generalizable graph learning via stochastic attention mechanism. In *Proceedings of the 39th International Conference on Machine Learning*, pages 15524–15543, 2022.

M. Mørup and L. K. Hansen. Archetypal analysis for machine learning and data mining. *Neurocomputing*, 80:54–63, 2012. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2011.06.033. URL `https://www.sciencedirect.com/science/article/pii/S0925231211006060`. Special Issue on Machine Learning for Signal Processing 2010.

N. Nakis, A. Celikkanat, L. Boucherie, C. Djurhuus, F. Burmester, D. M. Holmelund, M. Frolcová, and M. Mørup. Characterizing polarization in social networks using the signed relational latent distance model. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistic*, pages 11489–11505, 2023a.

N. Nakis, A. Çelikkanat, S. L. Jørgensen, and M. Mørup. A hierarchical block distance model for ultra low-dimensional graph representations, 2023b. URL `https://arxiv.org/abs/2204.05885`.

N. Nakis, C. Kosma, A. Brativnyk, M. Chatzianastasis, I. Evdaimon, and M. Vazirgiannis. The signed two-space proximity model for learning representations in protein–protein interaction networks. *Bioinformatics*, 41(6):btaf204, 2025a.

N. Nakis, C. Kosma, G. Nikolentzos, M. Chatzianastasis, I. Evdaimon, and M. Vazirgiannis. Signed graph autoencoder for explainable and polarization-aware network embeddings. In *Proceedings of the 28th International Conference on Artificial Intelligence and Statistic*, pages 496–504, 2025b.

B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 701–710, 2014.

B. Perozzi, V. Kulkarni, H. Chen, and S. Skiena. Don't walk, skip! online learning of multi-scale network

embeddings. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 258–265, 2017.

J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, and J. Tang. Network embedding as matrix factorization: Unifying DeepWalk, LINE, PTE, and Node2Vec. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*, pages 459–467, 2018.

A. Ragno, B. La Rosa, and R. Capobianco. Prototype-based interpretable graph neural networks. *IEEE Transactions on Artificial Intelligence*, 5(4):1486–1495, 2022.

M. T. Ribeiro, S. Singh, and C. Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144, 2016.

R. A. Rossi and N. K. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, 2015. URL `https://networkrepository.com`.

B. Rozemberczki and R. Sarkar. Characteristic functions on graphs: Birds of a feather, from statistical descriptors to parametric models, 2020. URL `https://arxiv.org/abs/2005.07959`.

C. Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5): 206–215, 2019.

P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. *AI magazine*, 2008.

G. Serra and M. Niepert. L2xgnn: learning to explain graph neural networks. *Machine Learning*, 113(9): 6787–6809, 2024.

O. Shchur and S. Günnemann. Overlapping community detection with graph neural networks. *arXiv preprint arXiv:1909.12201*, 2019.

O. Shoval, G. Shinar, Y. Hart, O. Ramote, A. Mayo, E. Dekel, K. Kavanagh, and U. Alon. Evolutionary trade-offs, pareto optimality, and the geometry of phenotype space. *Science (New York, N.Y.)*, 336: 1157–60, 04 2012. doi: 10.1126/science.1217405.

B.-J. Sun, H. Shen, J. Gao, W. Ouyang, and X. Cheng. A non-negative symmetric encoder-decoder approach for community detection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 597–606, 2017.

H. Sun, F. He, J. Huang, Y. Sun, Y. Li, C. Wang, L. He, Z. Sun, and X. Jia. Network embedding for community detection in attributed networks. *ACM Transactions on Knowledge Discovery from Data*, 14 (3):1–25, 2020.

F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu. Learning deep representations for graph clustering. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, pages 1293–1299, 2014.

A. Tsitsulin, J. Palowitch, B. Perozzi, and E. Müller. Graph clustering with graph neural networks, 2023. URL `https://arxiv.org/abs/2006.16904`.

M. N. Vu and M. T. Thai. Pgm-explainer: probabilistic graphical model explanations for graph neural networks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pages 12225–12235, 2020.

C. Wang, S. Pan, G. Long, X. Zhu, and J. Jiang. Mgae: Marginalized graph autoencoder for graph clustering. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 889–898, 2017a.

C. Wang, S. Pan, R. Hu, G. Long, J. Jiang, and C. Zhang. Attributed graph clustering: a deep attentional embedding approach. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 3670–3676, 2019.

H. Wang, Y. Zhang, Z. Zhao, Z. Cai, X. Xia, and X. Xu. Simple yet effective heuristic community detection with graph convolution network. *Scientific Reports*, 15(1), Nov. 2025. ISSN 2045-2322. doi: 10.1038/s41598-025-22860-z. URL `http://dx.doi.org/10.1038/s41598-025-22860-z`.

X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang. Community preserving network embedding. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 2017b.

X. Wang, J. Li, L. Yang, and H. Mi. Unsupervised learning for community detection in attributed networks based on graph convolutional network. *Neurocomputing*, 456:147–155, 2021.

Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2020.

Y. Xie, X. Wang, D. Jiang, and R. Xu. High-performance community detection in social networks using a deep transitive autoencoder. *Information Sciences*, 493:75–90, 2019.

L. Yang, X. Cao, D. He, C. Wang, X. Wang, and W. Zhang. Modularity based community detection with deep learning. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pages 2252–2258, 2016.

R. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec. Gnnexplainer: generating explanations for graph neural networks. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 9244–9255, 2019.

H. Yuan, J. Tang, X. Hu, and S. Ji. Xgnn: Towards model-level explanations of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 430–438, 2020.

H. Yuan, H. Yu, J. Wang, K. Li, and S. Ji. On explainability of graph neural networks via subgraph explorations. In *Proceedings of the 38th International Conference on Machine Learning*, pages 12241–12252, 2021.

Z. Zhang, P. Cui, H. Li, X. Wang, and W. Zhu. Billion-scale network embedding with iterative random projection. In *Proceedings of the 2018 IEEE International Conference on Data Mining*, pages 787–796, 2018.

Z. Zhang, Q. Liu, H. Wang, C. Lu, and C. Lee. Protgnn: Towards self-explaining graph neural networks. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence*, pages 9127–9135, 2022.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. Yes
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. Yes
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. Yes as a supplementary file

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. Yes
   (b) Complete proofs of all theoretical results. Yes in the supplementary
   (c) Clear explanations of any assumptions. Yes

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). Yes in the supplementary
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). Yes
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). Yes
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). Yes

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. Yes
   (b) The license information of the assets, if applicable. Not Applicable
   (c) New assets either in the supplemental material or as a URL, if applicable. Not Applicable
   (d) Information about consent from data providers/curators. Not Applicable
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. Not Applicable

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. Not Applicable
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. Not Applicable
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. Not Applicable