CHyBCE: A novel Clustering technique with Hyperbolic based Community Embedding for Community Detection

Ibtissam Fatma Chouchou[1], Ines Manel Boumazouza[1], Mohamed Chakib Bourzag[1], Merwan Bekkar[1], Ramzi Baaguigui[1], Marouane Benbetka[1], Ahmed Rayane Kebir[1], Fatima Benbouzid-Si Tayeb[12] and Malika Bessedik[12]

[1]*Ecole Nationale Superieure d'Informatique (ESI), BP 68M - 16270 Oued Smar, Alger, Algeria*

[2]*Laboratoire des Methodes de Conception de Systemes (LMCS), BP 68M - 16270 Oued Smar, Alger, Algeria*

Abstract

Community detection in social networks is a crucial task for understanding the intricate dynamics of interactions among individuals and groups. Traditional methods primarily focus on Euclidean graph embedding for community detection. However, recent advancements have shown that hyperbolic representations can more accurately capture the hierarchical and complex structures inherent in social networks. In this paper, we propose a novel approach that embeds graphs into hyperbolic space using Riemannian geometry, followed by applying the X-medoids clustering algorithm in the same hyperbolic space. Our methodology leverages the advantages of hyperbolic embeddings to improve the accuracy and efficiency of community detection. We validate the effectiveness of our approach through extensive experiments on both synthetic and real-world social network datasets, demonstrating superior performance compared to existing Euclidean-based methods.

Key words. Community detection, Social Network Analysis, hyperbolic embedding, Riemannian geometry, Clustering.

1. INTRODUCTION

In recent years, social media has become an integral part of daily life, with over five billion users worldwide in 2024, a number projected to increase to over six billion by 2028 [30]. This massive usage of social networks provides a rich dataset for analyzing and understanding social dynamics, particularly through the detection of communities within these networks.

Community detection, a core problem in social network analysis, involves identifying distinct groups of nodes within a network characterized by robust interconnections. This is achieved by effectively partitioning network nodes into cohesive clusters with dense internal connections while maintaining sparse connections with nodes in other clusters.

Various approaches have been proposed to address the community detection problem in social networks. Some of these approaches are known as traditional algorithms, including graph partitioning [14], hierarchical clustering [14, 15], spectral clustering [14, 13], and modularity optimization techniques [25, 12, 20]. Graph partitioning divides the network into optimal segments, while hierarchical clustering constructs a cluster hierarchy using either agglomerative (bottom-up) or divisive (top-down) methods. Spectral clustering reduces dimensions using eigenvalues of the graph's Laplacian matrix before clustering, and modularity optimization maximizes internal link density within communities compared to links between different communities. The Louvain algorithm [12] is particularly noteworthy among these approaches.

Detecting communities in intricate networks presents significant challenges due to their growing complexity. Traditional methods, while effective for simpler networks, often fail to partition these more elaborate structures accurately. This is where machine learning (ML) stands out, given its ability to manage high-dimensional data and streamline data processing, making it particularly suited for addressing the complexities of modern social networks.

In the state-of-the-art, two primary ML approaches for community detection are distinguished. The first applies pure clustering techniques directly to graphs, such as [31, 37, 22, 36]. The second approach, consists of Euclidean clustering after Euclidean embedding, involves embedding the graph data into Euclidean space using techniques like Node2Vec, Graph2Vec, DeepWalk [27], and Extreme Learning Machine [34], followed by traditional clustering methods like K-Means and k-medoids. This approach has been notably successful in various studies.

Embedding-based methods are trending due to their ability to transform complex network data into lower-dimensional spaces where traditional clustering techniques can be applied more effectively. However, Euclidean embeddings have notable drawbacks. They require large dimensions to capture certain complex relationships, such as hierarchical structures, which can lead to inefficiencies and higher computational costs.

Recently, hyperbolic embeddings have emerged as a superior alternative [17], especially for representing complex, hierarchical data. Unlike Euclidean spaces, hyperbolic spaces can capture intricate relationships with lower dimensions. Studies have shown that hyperbolic embeddings, such as the Poincaré disk model, can effectively represent hierarchical structures with just two dimensions, providing clearer visualizations and more efficient clustering.

Motivated by the success of hyperbolic embeddings and the potential of hyperbolic learning algorithms, we propose a new approach for learning node and community representations on graphs. Our contribution involves embedding the graph into Riemannian hyperbolic space [17] and applying an X-medoids clustering algorithm within this space, using the silhouette metric adapted for hyperbolic geometry.

The remainder of this paper is organized as follows: in Section 2 defines the problem of community detection in social networks. In Section 3, we review the geometry of the Poincaré ball and related tools, such as Riemannian barycenters and Riemannian Gaussian distributions, and present our proposed approach. Section 4 outlines the experiments conducted to evaluate the effectiveness of our method, along with the obtained results and their interpretation. Finally, Section 5 concludes the paper with our findings and proposes potential avenues for future research.

2. PROBLEM STATEMENT

This section sets the stage for discussing the specifics of community detection, outlining its goals and the methodologies used to achieve them.

A social network can be modeled by a graph $G = (V, E)$, which includes $N = |V|$ nodes and $m = |E|$ edges. Here, V represents the set of nodes and E represents the set of edges. Formally, $V = \{v_1, v_2, \ldots, v_n\}$ and $E = \{e_{ij}\}_{i,j=1}^{n}$, where e_{ij} indicates an edge between nodes v_i and v_j.

The task of community detection involves devising a methodology to identify communities within a given network, where nodes correspond to users and edges represent connections between these users. The similarity w_{ij} between users i and j is usually quantified by the weight of the edge connecting i and j. This weight lies within the interval $[0, 1]$ and reflects the degree of similarity in their interests. For our case, we will only be interested in the topological aspect of the graph so the weights are represented by the adjacency matrix instead (1 if neighbors, 0 otherwise).

The outcome of community detection is a partition of the network into a set of communities $C = \{C_1, C_2, \ldots, C_K\}$. A node v_i assigned to a community C_k must satisfy the condition that its internal degree within the community C_k is greater than its external degree with nodes outside C_k.

If $C_k \cap C_{k'} = \emptyset$ for all $k \neq k'$, the communities C are considered disjoint. However, if there exist nodes that belong to multiple communities, i.e., $C_k \cap C_{k'} \neq \emptyset$ for some $k \neq k'$, the communities are considered overlapping.

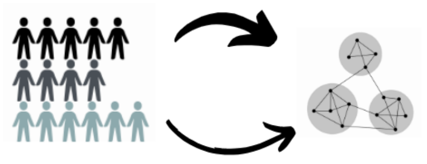

Fig. 1: Community detection process.

The objective of community detection is thus to find a partition or cover of the graph that optimizes the internal cohesion of communities while minimizing the external connections. Various algorithms and techniques have been proposed to address this problem, each with different assumptions and optimization criteria, reflecting the complexity and diversity of real-world networks.

3. PROPOSED SOLUTION

In this section, we introduce our novel approach for embedding and analyzing community structures within hyperbolic spaces. *Section 3.1: Hyperbolic Space: A Motivation* discusses the advantages of hyperbolic geometry for complex networks. *Section 3.2: Hyperbolic Space: Theoretical Toolbox* provides essential concepts and tools, including the Poincaré ball model. *Section 3.3: Hyperbolic Community Embeddings* details our methodology for embedding community structures in hyperbolic space. Finally, *Section 3.4: Clustering with X-Medoids* describes clustering the embedded nodes using the X-Medoids algorithm.

3.1. Hyperbolic Space: A motivation

Graph embedding methods aim to learn representations of nodes that preserve graph properties, such as graph distances. These embeddings can be utilized in downstream applications, including clustering, visualization, nearest neighbor search, and classification tasks. Many solutions adopt an embedding-based approach, transferring data from one metric space to another that is easier to work with, while preserving the information present in the original metric space.

Most machine learning algorithms traditionally learn embeddings in standard Euclidean spaces. Solutions like Node2Vec, DeepWalk, and LINE [32], for example, use Euclidean embedding techniques to map graph data into a space where typical machine learning models can be applied effectively. These methods have been widely adopted due to their straightforward implementation and well-understood mathematical properties.

However, Euclidean space is limited in its ability to express certain types of graph-related data. Specifically, the Euclidean space may fail to capture the inherent hierarchical or tree-like structure of many real-world graphs. An illustrative example, as noted in the thesis by Ines Chami [11], shows that trees cannot be embedded perfectly into Euclidean spaces of any dimension, meaning that the structural nuances of tree-like graphs are often lost. Recent research highlights the promise of non-Euclidean geometries, such as hyperbolic or spherical geometries, for more faithful embeddings. Hyperbolic space, in particular, has shown significant potential in embedding graph nodes. This space can conserve a maximum amount of information from the original graph, providing a richer and more expressive representation compared to Euclidean space.

For instance, trees can be embedded almost perfectly into two-dimensional hyperbolic spaces, capturing their hierarchical structure in a way that Euclidean spaces cannot. This is because hyperbolic space can naturally represent exponential growth, which is characteristic of tree structures, thereby maintaining the relative distances between nodes more accurately.

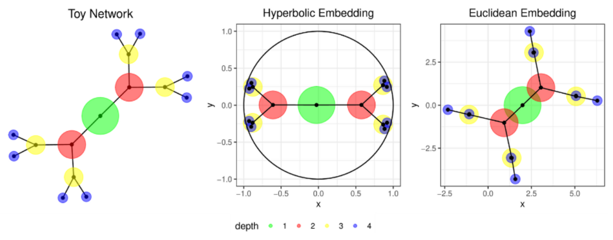

Fig. 2: Tree embedding in euclidian space vs Tree embedding in hyperbolic space.

By embedding graph data into hyperbolic space, we can achieve embeddings that retain more of the original graph's properties, leading to improved performance in various downstream tasks. This shows that for complex graph structures, hyperbolic embeddings provide a powerful alternative to traditional Euclidean embeddings, ensuring a more accurate and expressive representation of the graph data.

3.2. Hyperbolic Space: Theoretical Toolbox

In this subsection, we delve into the intricacies of the Poincaré ball model, a hyperbolic space characterized by a Riemannian manifold with constant negative sectional curvature. We begin by exploring the fundamental properties of this space, including the expressions for the exponential and logarithmic maps, which play a crucial role in our subsequent discussions.

Furthermore, we examine the concepts of Riemannian barycentre and Riemannian Gaussian distributions, elucidating their definitions and key properties. These concepts serve as foundational elements for the algorithms we employ in the forthcoming sections, namely Expectation-Maximization. By leveraging these fundamental notions, our approach aims to unlock the full potential of hyperbolic geometry in solving complex problems in machine learning and data analysis.

3.2.1. Geometry of the Poincaré ball

The Poincaré ball model of m dimensions $(\mathbb{B}^m, g^{B^m})$ is the manifold $\mathbb{B}^m = \{x \in \mathbb{R}^m : ||x|| < 1\}$ equipped with the Riemannian metric:

$$g_x^{\mathbb{B}^m} = \frac{4}{(1-||x||^2)^2} g^{\mathbb{E}}$$

where $g^{\mathbb{E}}$ is the Euclidean scalar product. The Riemannian distance between two points $x, y \in \mathbb{B}^m$, induced by this metric, is given by:

$$d(x,y) = \operatorname{arcosh}\left(1 + 2\frac{||x-y||^2}{(1-||x||^2)(1-||y||^2)}\right)$$

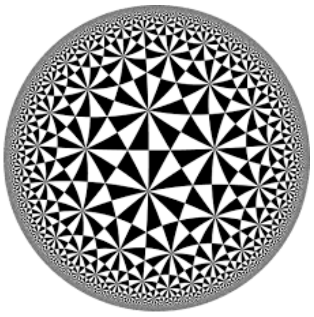

Fig. 3: Representation of the Poincarré Ball.

Equipped with this metric, $\mathbb{B}^m$ becomes a hyperbolic space [18]. The Riemannian exponential and logarithmic maps associated with this distance are explicitly given by [16]. First, the Möbius addition operator $\oplus$ for $x, y \in \mathbb{B}^m$ is defined as:

$$x \oplus y = \frac{(1 + 2\langle x, y\rangle + \|y\|^2)x + (1 - \|x\|^2)y}{1 + 2\langle x, y\rangle + \|x\|^2\|y\|^2}$$

For $x \in B^m$ and $y \in \mathbb{R}^m \setminus \{0\}$, the exponential map is defined as:

$$\mathrm{Exp}_x(y) = x \oplus \left(\tanh\left(\frac{\|y\|}{1-\|x\|^2}\right)\frac{y}{\|y\|}\right)$$

Intuitively, the exponential map is a generalization to manifolds of the addition between a point and a vector. Adding a point x with a tangent vector y at x results in a point z belonging to the geodesic initiated from x and following the tangent vector. Exp_x is a diffeomorphism from $\mathbb{R}^m$ to $\mathbb{B}^m$. Its inverse, called the logarithmic map, is defined for all $x \neq y \in \mathbb{B}^m$ as:

$$\mathrm{Log}_x(y) = \frac{(1-\|x\|^2)\tanh^{-1}(\|-x \oplus y\|)}{\|-x \oplus y\|}(-x \oplus y)$$

The logarithmic map serves as a generalized subtraction operation for manifolds. When given two points x and y, it yields the tangent vector originating from x and pointing towards y. Figure 2 provides a visual representation of the Riemannian logarithmic and exponential maps on a manifold.

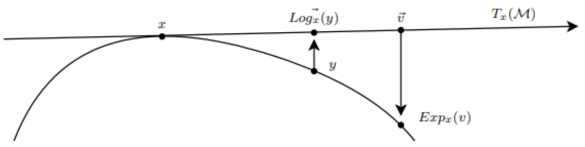

Fig. 4: Representation of the logarithmic and exponential maps on a manifold M: Given two points x, y M, Logx(y) is a tangent vector at x pointing towards y; and given some tangent vector v at x, then Expx(v) is the point on the geodesic initiated from x that follows v. [17]

In addition, parallel transport is a fundamental concept in optimization on Riemannian manifolds. It facilitates the transfer of a tangent vector from one point to another along a geodesic curve. In the context of the Poincaré ball, the parallel transport of a vector v from point x to y is defined as:

$$\phi_{x\to y}(v) = \mathrm{gyration}(y, -x, v)\frac{\lambda_x}{\lambda_y}$$

where $\lambda_x = \frac{2}{1-||x||^2}$ represents the conformal factor, and the gyration function is detailed in [16, 33]. Parallel transport plays a crucial role in optimization techniques employing adaptive methods such as RAdam or RAMSGrad.

3.2.2. Riemannian barycentre

In the realm of differential geometry, the concept of a Riemannian barycenter, or Fréchet mean, is crucial for understanding the central tendency of a set of points on a manifold. For a Riemannian manifold with negative curvature, such as the Poincaré ball model $\mathbb{B}^m$, the existence and uniqueness of the Riemannian barycenter are guaranteed [2].

Given a set of points $\{x_i \mid 1 \leq i \leq n\}$ in $\mathbb{B}^m$, the empirical Riemannian barycenter $\hat{\mu}_n$ is defined as the point that minimizes the sum of squared Riemannian distances to each point in the set:

$$\hat{\mu}_n = \arg\min_{\mu \in \mathbb{B}^m} \sum_{i=1}^{n} d^2(\mu, x_i),$$

where $d(\mu, x_i)$ denotes the Riemannian distance between μ and x_i.

The computation of the Riemannian barycenter is essential in various applications, such as averaging shapes, clustering on manifolds, and intrinsic statistical analysis on non-Euclidean spaces. To approximate $\hat{\mu}_n$ numerically, several stochastic gradient algorithms have been proposed [7, 3, 4, 5, 6], which iteratively adjust a candidate point by considering the gradient of the objective function in the Riemannian space. These algorithms leverage the manifold's geometry to ensure convergence to the unique barycenter.

This concept is particularly relevant in fields like machine learning and computer vision, where data often resides on curved spaces rather than flat Euclidean spaces. The ability to compute a central point that respects the underlying geometry of the data is invaluable for tasks such as clustering, dimensionality reduction, and data summarization.

3.2.3. Riemannian Gaussian Distributions

In the context of Riemannian geometry, particularly on negatively curved manifolds such as the Poincaré ball model $\mathbb{B}^m$, the concept of Riemannian Gaussian distributions is an essential extension of classical Gaussian distributions. These distributions are defined in a way that respects the underlying geometric structure of the manifold.

The probability density function of a Riemannian Gaussian distribution on the Poincaré ball is given by:

$$f(x \mid \mu, \sigma) = \frac{1}{\zeta_m(\mu, \sigma)} \exp\left(-\frac{d^2(x, \mu)}{2\sigma^2}\right),$$

where $d(x, \mu)$ denotes the Riemannian distance between the point x and the mean μ, and σ represents the standard deviation.

The normalization coefficient $\zeta_m(\mu, \sigma)$ ensures that the probability density integrates to one over the manifold. It is given by:

$$\zeta_m(\mu, \sigma) = \zeta_m(0, \sigma) = \zeta_m(\sigma) = \int_{B^m} \exp\left(-\frac{d^2(x, 0)}{2\sigma^2}\right) dv(x)$$

where $dv(x)$ denotes the Riemannian volume element. An explicit formula for the normalization coefficient $\zeta_m(\sigma)$ is:

$$\zeta_m(\sigma) = \sqrt{\frac{\pi}{2}} \frac{\sigma}{2^{m-1}} \sum_{k=0}^{m-1} (-1)^k C_k^{m-1} e^{\frac{p_k^2 \sigma^2}{2}} \left(1 + \operatorname{erf}(\frac{p_k \sigma}{\sqrt{2}})\right)$$

with $p_k = (m-1) - 2k$ and C_k^{m-1} being the binomial coefficients.

For parameter estimation, the maximum likelihood estimate (MLE) of the mean μ is the Riemannian barycenter $\hat{\mu}_n$ of the sample points $x_1, \ldots, x_n$:

$$\hat{\mu}_n = \text{Riemannian Barycentre}(x_1, \ldots, x_n).$$

The MLE of the standard deviation σ is given by:

$$\hat{\sigma}_n = \Phi\left(\frac{1}{n} \sum_{i=1}^{n} d^2(\hat{\mu}_n, x_i)\right),$$

where $\Phi : \mathbb{R}^+ \to \mathbb{R}^+$ is a strictly increasing bijective function defined as the inverse of:

$$\sigma \mapsto \sigma^3 \times \frac{d}{d\sigma} \log \zeta_m(\sigma).$$

These Riemannian Gaussian distributions are fundamental for probabilistic modeling on manifolds, enabling tasks such as classification, clustering, and hypothesis testing to be performed in a geometrically consistent manner. We will be using them with Gaussian Mixture Models (GMMs) which will be presented in the upcoming subsection.

3.2.4. Hyperbolic Gaussian Mixture Model

The HGMM consists of K components, each being a Riemannian Gaussian distribution on the Poincaré ball. The probability density function (pdf) of the HGMM is given by:

$$F(x \mid \Theta) = \sum_{k=1}^{K} \pi_k f_k(x \mid \mu_k, \sigma_k),$$

where:

- $x \in \mathbb{B}^m$ is a point in the hyperbolic space,
- π_k are the mixture coefficients satisfying $\sum_{k=1}^{K} \pi_k = 1$ and $\pi_k \geq 0$,
- $f_k(x \mid \mu_k, \sigma_k)$ is the pdf of the k-th Riemannian Gaussian component:
$$f_k(x \mid \mu_k, \sigma_k) = \frac{1}{\zeta_m(\sigma_k)} \exp\left(-\frac{d^2(x, \mu_k)}{2\sigma_k^2}\right),$$
- $\mu_k \in \mathbb{B}^m$ is the mean of the k-th component,
- $\sigma_k > 0$ is the standard deviation of the k-th component.
- $\zeta_m(\sigma_k)$ is the normalization coefficient.

Parameter Estimation via EM Algorithm

The Riemannian Expectation-Maximization (EM) algorithm, similar to its Euclidean counterpart, is used to estimate the parameters of the HGMM. It involves iteratively performing Expectation (E) and Maximization (M) steps.

E-Step

In the E-step, we compute the responsibilities γ_{ik}, which represent the probability of each point x_i belonging to each component k:

$$\gamma_{ik} = P(z_i = k \mid x_i) = \frac{\pi_k f_k(x_i \mid \mu_k, \sigma_k)}{\sum_{i=1}^{N} \pi_k f_k(x_i \mid \mu_k, \sigma_k)}. \quad (1)$$

In this expression $z_1, \ldots, z_N$ represent the latent variables associated with the mixture model.

M-Step

In the M-step, we update the parameters π_k, μ_k, and σ_k:

Mixture Coefficients The mixture coefficients represent the weights of each gaussian distribution and are updated as:

$$\pi_k = \frac{1}{N} \sum_{i=1}^{N} \gamma_{ik}. \quad (2)$$

Mean Updating the mean μ_k involves computing the weighted Riemannian barycentre of the points assigned to each component:

$$\hat{\mu}_k = \arg\min_{\mu} \sum_{i=1}^{N} \gamma_{ik} d^2(\mu, x_i) \quad (3)$$

using Riemannian optimization. The algorithm from [3] is employed to provide an estimate for $\hat{\mu}_k$.

Algorithm 1 Weighted Barycenter Calculation

```
Require: $W = (\gamma_{ik})$ weight matrix, $\{x_1, \ldots, x_N\}$ a subset of $\mathbb{B}^m$, $\epsilon$ small convergence rate, $\lambda$ learning rate for the barycenter
1: Initialize $\mu_k^0$
2: repeat
3:     $\mu_k^{t+1} \leftarrow \mathrm{Exp}_{\mu_k^t}\left(\frac{\lambda}{2\sum_{i=1}^{N}\gamma_{ik}} \sum_{i=1}^{N} \gamma_{ik} \mathrm{Log}_{\mu_k^t}(x_i)\right)$
4: until $\|\mu_k^t - \mu_k^{t+1}\| \leq \epsilon$
5: return $\mu_k^{t+1}$
```

Standard Deviation

The maximum likelihood estimate (MLE) for the standard deviation σ_k of the Gaussian distribution, as discussed in Section 3.2.3, is obtained by solving:

$$\hat{\sigma}_k = \arg\min_{\sigma_s} \left| \frac{1}{\sum_{i=1}^{N} \gamma_{ik}} \sum_{i=1}^{N} d^2(\mu_k, x_i)\gamma_{ik} - \Phi^{-1}(\sigma_s) \right| \quad (4)$$

A grid-search approach is utilized to approximate σ_k by evaluating it for a discrete set of σ_s. To compute $\Phi^{-1}(\sigma_s)$, which includes the term $\frac{d}{d\sigma} \log \zeta_m(\sigma)$, we use an automatic differentiation algorithm provided by the back-end system.

3.3. Hyperbolic community embeddings

Let's define a graph $G(V, E)$ where V is the set of nodes and $E \subset V \times V$ is the set of edges. The goal is to provide a faithful and exploitable representation of the graph structure. it is mainly achieved using different proximity measures presented as follows:

First-order proximity. It captures the local pairwise similarity between nodes. It ensures that directly connected nodes (i.e., nodes with an edge between them) have similar embeddings. This is achieved by minimizing the distance between the embeddings of nodes that are directly connected. It is preserved by optimizing a loss function similar to [26]:

$$O_1 = - \sum_{(v_i, v_j) \in E} \log(\sigma(-d^2(\phi_i, \phi_j))) \quad (5)$$

with $\sigma(x) = \frac{1}{1+e^{-x}}$ being the sigmoid function, $\phi_i \in \mathbb{B}^m$ is the embedding of the i-th node of V, and d is the Riemannian distance.

Second-order proximity. It captures the similarity between nodes based on their shared neighbors. It ensures that nodes with similar neighborhoods have similar embeddings, even if they are not directly connected. This is typically achieved through techniques like negative sampling [24], where the embeddings of a node are optimized to be close to the embeddings of its context nodes (neighbors) and distinct from non-context nodes. For that we considered the loss:

$$O_2 = - \sum_{v_i \in V} \sum_{v_j \in C_i} \Big[\log(\sigma(-d^2(\phi_i, \phi'_j))) + \sum_{v_k \sim P_n} \log(\sigma(d^2(\phi_i, \phi'_k))) \Big] \quad (6)$$

with C_i being the nodes in the context of the i-th node, $\phi'_j \in \mathbb{B}^m$ the embedding of $v_j \in C_i$, and P_n the negative sampling distribution over V given by $P_n(v) = \frac{\deg(v)^{3/4}}{\sum_{v_i \in V} \deg(v_i)^{3/4}}$. The next paragraph introduces a third objective function allowing to detect and embed communities rather than individual nodes.

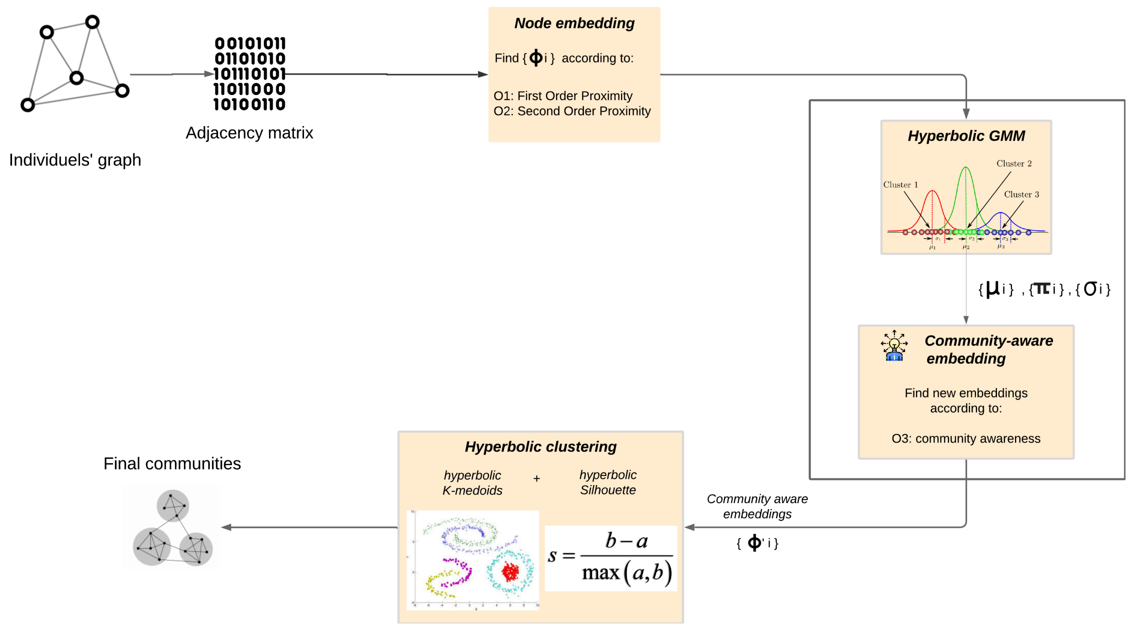

Fig. 5: CHyBCE Global Architecture.

Community-aware embedding Our primary goal is to learn embeddings for community detection and classification. To achieve this, it is essential to ensure that nodes embedded within the same community are closely positioned. In alignment with the approach described in ComE [9] and following the one used in [17], we integrate node embeddings (considering both first and second-order proximities through the minimization of O_1 and O_2) with community awareness by introducing a third-order loss function O_3. This third-order loss, referred to as the community loss, is expressed as follows:

$$O_3 = -\sum_{i=1}^{N}\sum_{k=0}^{K}\gamma_{ik}\log\left(\frac{1}{\zeta_m(\sigma_k)}e^{-\frac{d^2(x_i,\mu_k)}{2\sigma_k^2}}\right) \tag{7}$$

To jointly solve for community and graph embeddings, we minimize the combined loss function as presented in [17]:

$$L = \alpha O_1 + \beta O_2 + \gamma O_3 \tag{8}$$

with α, β, γ the weights of respectively first, second-order, and community losses.

3.3.1. *Optimization*

We now focus on the Riemannian optimization of the loss functions O_1, O_2, and O_3. Following the approach in [16], we employ the Riemannian Gradient Descent (RGD) Algorithm. This algorithm first computes the gradient in the tangent space and then projects the updated values onto the manifold using the exponential map:

$$\phi_{t+1} = \exp_{\phi_t}\left(-\eta\frac{\partial O}{\partial \phi}\right) \tag{9}$$

where O is the function to optimize, ϕ is a parameter, $t \in \{1, 2, \ldots\}$ is the iteration number, and η is the learning rate. As described in [3], the gradient of the distance $d^p(\phi_i, \phi_j)$ where $\phi_i, \phi_j \in B^m$ is given by:

$$\nabla_{\phi_i} d_p(\phi_i, \phi_j) = -p \cdot d^{p-1}(\phi_i, \phi_j) \cdot \frac{\log_{\phi_i}(\phi_j)}{d(\phi_i, \phi_j)} \tag{10}$$

Using the chain rule, the gradient of a function $h = g \circ d_p$ (where g is a differentiable function) can be computed as:

$$\nabla_{\phi_i} h = g'(d_p(\phi_i, \phi_j))\nabla_{\phi_i} d_p(\phi_i, \phi_j) \tag{11}$$

where the expression for $\nabla_{\phi_i} d_p(\phi_i, \phi_j)$ is given in the equation above. Optimization in this implementation was performed by redefining the gradient of the distance and then using standard auto-differentiation tools provided by the PyTorch backend.

3.4. Clustering with X-Medoids

The Hyperbolic X-Medoids (HX-M) algorithm is an adaptation of the traditional k-medoids algorithm for clustering [28] in hyperbolic space. Unlike k-means, which computes the barycentre of the points in a cluster, k-medoids chooses as cluster centers (medoids) actual points whose the sum of distances to the points in the same cluster is minimal (as shown in 2). Additionally, the number of clusters (communities) K is determined based on the global clustering quality.

Algorithm Principles The key principles of the HX-M algorithm are:

- **Medoids Selection:** The centroids (medoids) are chosen from the actual data points.
- **Hyperbolic Distance:** Distances between points are computed using hyperbolic geometry.
- **Automatic Cluster Number Determination:** The algorithm automatically determines the optimal number of clusters using the hyperbolic silhouette score.

Hyperbolic Silhouette Score To determine the suitable number of clusters, we redefine the traditional silhouette score for hyperbolic space. The silhouette score measures how similar a point is to its own cluster compared to other clusters. In hyperbolic space, the distance metric d_H is used.

The silhouette score for a point i is defined as:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

where:

- $a(i)$ is the average distance from point i to all other points in the same cluster.
- $b(i)$ is the minimum average distance from point i to all points in any other cluster.

Hyperbolic X-Medoids Algorithm The HX-M algorithm effectively clusters data in hyperbolic space by leveraging the properties of medoids and an automatic cluster determination method based on the hyperbolic silhouette score as shown in 3. This approach is particularly useful for data exhibiting a hyperbolic geometry.

Algorithm 2 Hyperbolic K-Medoids

```
1: Input: $\phi \subset \mathbb{B}^{n\times m}$ (node embeddings), max_iter (maximum iterations)
2: Output: $I$ (labels of each sample), $\mu$ (cluster medoids)
3: Initialize medoids $\mu \subset \mathbb{B}^{K\times m}$ randomly from the data points
4: for $t \in \{1, 2, \ldots, \max\_iter\}$ or convergence do
5:     for $i \in \{1, 2, \ldots, n\}$ do
6:         $I_i^t \leftarrow \arg\min_j d_H(\mu_j, \phi_i)$
7:     end for
8:     for $k \in \{1, 2, \ldots, K\}$ do
9:         Update medoid
10:        $\mu_k \leftarrow \arg\min_{\phi_j \in \{\phi_i | I_i = k\}} \sum_{\phi_i \in \{\phi_j | I_j = k\}} d_H(\phi_i, \phi_j)$
11:    end for
12:    convergence $\leftarrow$ True iff $\forall i \in \{1, 2, \ldots, n\}, I_i^{t-1} = I_i^t$
13: end for
```

Algorithm 3 Hyperbolic X-Medoids

```
1: Input: $\phi \subset \mathbb{B}^{n\times m}$ (node embeddings), min_k, max_k, max_iter (maximum iterations)
2: Output: $I$ (labels of each sample), $\mu$ (cluster medoids)
3: Initialize best silhouette score $s_{\text{best}} \leftarrow -1$
4: Initialize best clusters $I_{\text{best}} \leftarrow \emptyset$, best medoids $\mu_{\text{best}} \leftarrow \emptyset$
5: for $K \in \{\min\_k, \min\_k + 1, \ldots, \max\_k\}$ do
6:     Initialize medoids $\mu \subset \mathbb{B}^{K\times m}$ randomly from the data points
7:     for $t \in \{1, 2, \ldots, \max\_iter\}$ or convergence do
8:         for $i \in \{1, 2, \ldots, n\}$ do
9:             $I_i^t \leftarrow \arg\min_j d_H(\mu_j, \phi_i)$
10:        end for
11:        for $k \in \{1, 2, \ldots, K\}$ do
12:            Update medoid $\mu_k \leftarrow$
13:            $\arg\min_{\phi_j \in \{\phi_i | I_i = k\}} \sum_{\phi_i \in \{\phi_j | I_j = k\}} d_H(\phi_i, \phi_j)$
14:        end for
15:        convergence $\leftarrow$ True iff $\forall i \in \{1, 2, \ldots, n\}, I_i^{t-1} = I_i^t$
16:    end for
17:    Compute hyperbolic silhouette score $s_K$ for the current clustering
18:    if $s_K > s_{\text{best}}$ then
19:        $s_{\text{best}} \leftarrow s_K$
20:        $I_{\text{best}} \leftarrow I$
21:        $\mu_{\text{best}} \leftarrow \mu$
22:    end if
23: end for
24: Output $I_{\text{best}}$, $\mu_{\text{best}}$
```

4. Experimental Results

In this section, we present a comprehensive analysis of our proposed algorithm's performance for community detection. We outline the metrics used for evaluation, detail the parameter settings for our experiments, describe the benchmark datasets employed, and compare our results against various configurations and other well-known algorithms.

4.1. Metrics

For the problem of community detection, there are several metrics to determine the quality of a solution, including modularity, Normalized Mutual Information (NMI), conductance, and similarity [10]. In this work, we use NMI as the objective function to maximize, and we evaluate the quality of our solutions compared to the ground truth community assignment using the modularity metric .

4.1.1. Modularity

Modularity measures the strength of the division of a graph into communities. It aims to maximize the connectivity between nodes within the same community and minimize the connections between nodes in different communities. The modularity Q is calculated as follows:

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j) \qquad (12)$$

where:

- Q is the modularity value,
- m is the total number of edges in the network,
- A_{ij} is the element in the adjacency matrix corresponding to nodes i and j,
- k_i and k_j are the degrees of nodes i and j, respectively,
- c_i and c_j are the communities to which nodes i and j belong, respectively,
- $\delta(c_i, c_j)$ is the Kronecker delta function, which is equal to 1 if c_i is equal to c_j, and 0 otherwise.

Modularity values range between -1 and 1, with higher values indicating stronger community structure. Despite some criticisms, modularity remains a widely used and effective metric for evaluating community structures.

4.1.2. Normalized Mutual Information (NMI)

Normalized Mutual Information (NMI) assesses the similarity between two partitions based on the information they share. NMI values range from 0 (no mutual information) to 1 (perfect correlation). NMI is calculated as follows:

$$\mathrm{NMI}(C, K) = \frac{-2 \sum_{c \in C} \sum_{k \in K} n_{ck} \log\left(\frac{n_{ck} N}{n_c n_k}\right)}{\sum_{c \in C} n_c \log\left(\frac{n_c}{N}\right) + \sum_{k \in K} n_k \log\left(\frac{n_k}{N}\right)} \qquad (13)$$

where:

- C represents the true community assignment,
- K represents the predicted community assignment,
- n_{ck} is the number of nodes assigned to both community c and community k,
- n_c is the number of nodes assigned to community c,
- n_k is the number of nodes assigned to community k,
- N is the total number of nodes in the network.

NMI provides a robust measure for evaluating the correspondence between the predicted and true community structures.

4.2. Parameters Settings

Some parameters involved in the learning process are difficult to know a priori. Therefore, we performed several grid-searches to find the hyperparameters with the best cross-validation performances in terms of modularity, when running our algorithm. In particular, the parameter λ, the learning rate, c, the size of the context window, and t, the number of negative sampling nodes, appeared to be the most influential on the results.

For our algorithm, the parameters c and t were selected from the set $\{5,\ 10\}$, β and $\alpha \in \{0.1, 1\}$, $\gamma \in \{0.01, 0.1\}$, and λ from $[1\mathrm{e}^{-2},\ 1\mathrm{e}^{-4}]$. For all experiments, we generated, for each node, 10 random walks, each of length 80. For the first 10 epochs, embeddings were trained using only O_1 and O_2 and then using the complete loop.

4.3. Benchmark Datasets

We evaluated our algorithm on a variety of benchmark datasets to ensure comprehensive performance assessment. The datasets used in our experiments include:

4.3.1. Friendship Network of Zachary's Karate Club

The Friendship Network of Zachary's Karate Club consists of 34 nodes and 78 edges. The actual data are divided into two communities [19].

4.3.2. Bottlenose Dolphin Network

The Bottlenose Dolphin Network consists of 62 nodes and 159 edges. The actual data are divided into two communities [23].

4.3.3. Books Network about US Politics

The Books Network about US Politics contains 105 nodes and 441 edges. The actual data are divided into three communities [1].

4.3.4. Football Network

The Football Network consists of 115 nodes and 613 edges. The actual data are divided into 12 communities [19].

4.3.5. Email Network

The Email Network from a large European research institution consists of 1,005 nodes and 25,571 edges. The actual data are divided into 42 communities [29].

4.3.6. LFR Benchmark

The LFR Benchmark is a synthetic dataset designed to test community detection algorithms. It generates graphs with known community structures. For our experiments, we used graphs with varying sizes and community distributions [21].

Dataset	Nodes	Edges	Communities
Zachary's Karate	34	78	2
Dolphin Network	62	159	2
Books Network	105	441	3
Football Network	115	613	12
Email Network	1,005	25,571	42

Table 1: Summary of benchmark datasets used in the experiments.

4.4. Performance Analysis

To thoroughly evaluate the performance of our solution, we compare it with several other configurations. Each configuration eliminates certain components and evaluates the remaining process. The configurations tested are:

- **Configuration 1: GMM + X-medoids**
  In this configuration, we utilize Gaussian Mixture Models (GMM) for initial clustering and X-medoids for further refinement. This configuration eliminates the proximity-based methods.

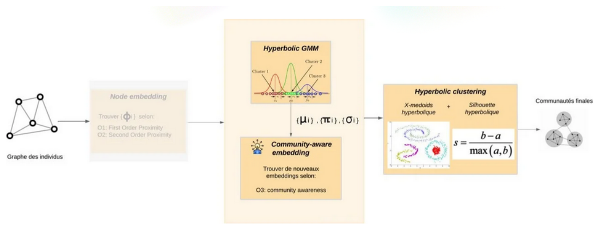

Fig. 6: Configuration 1: GMM + X-medoids.

- **Configuration 2: 1-2-proximity + X-medoids**
  This configuration uses first-order, second-order proximity measures combined with X-medoids for clustering, eliminating the K-means and 123-proximity methods.

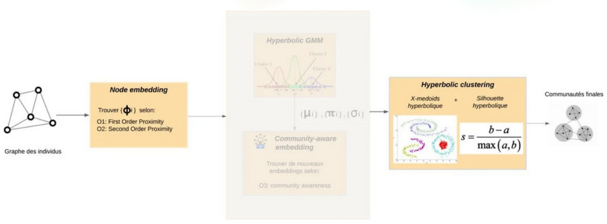

Fig. 7: Configuration 2: 1-2-proximity + X-medoids.

- **Configuration 3: 1-2-3-proximity + K-means**
  Here, first-order, second-order, and third-order proximity measures are employed alongside K-means clustering, eliminating the use of X-medoids.

- **Configuration 4: Our Solution**
  Our solution combines first-order, second-order, and third-order proximity measures with X-medoids.

Dataset	GMM + X-medoids		12-proximity + X-medoids		123-proximity + K-means		Our Solution	
	NMI	Modularity	NMI	Modularity	NMI	Modularity	NMI	Modularity
Zachary's Karate	0.83	0.35	0.83	0.35	**1**	**0.37**	**1**	**0.37**
Dolphin	0.75	0.38	0.65	0.38	0.70	0.38	**0.88**	0.37
Football	0.85	0.52	0.88	0.56	0.90	0.58	**0.94**	**0.60**
Polbooks	0.50	0.47	0.48	0.45	0.55	0.49	**0.67**	0.44
Email	0.54	0.29	0.67	0.35	0.69	0.34	**0.69**	0.34

Table 2: Performance metrics (NMI/Modularity) for different configurations across benchmark datasets.

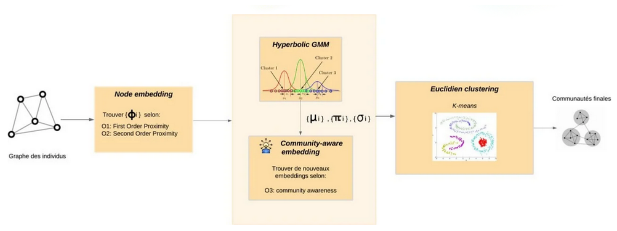

Fig. 8: Configuration 3: 1-2-3-proximity + K-means.

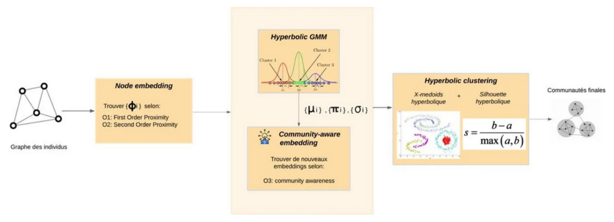

Fig. 9: Configuration 4: Our Solution.

Our experimental results demonstrate that each element of our architecture contributes effectively to the overall performance. By evaluating different configurations, we ensure that our solution consistently outperforms other methods in terms of both NMI and modularity across various benchmark datasets.

4.5. Comparison with Other Algorithms

To provide a comprehensive evaluation, we compared our solution against several well-known community detection algorithms: Louvain, Shallow Graph Convolutional Networks (SGCN), and ComE+.

Louvain Algorithm [12] The Louvain algorithm is a widely-used method for community detection in large networks. It optimizes the modularity of the partitions through a two-phase approach. In the first phase, it assigns each node to its own community and then iteratively merges communities to increase modularity. In the second phase, it aggregates nodes in the same community and treats each community as a single node, repeating the process until modularity cannot be improved. The Louvain algorithm is efficient and often produces high-quality community structures, making it a popular baseline for comparison.

Shallow Graph Convolutional Networks (SGCN) [35] The proposed Shallow-GCN (SGCN) method addresses the challenge of community detection in unlabeled graphs. SGCN begins by using the Structural Center Location (SCL) measure to assign initial labels to central nodes. These labels are then propagated to neighboring nodes, forming a labeled subset for training the GCN model. For graphs without node attributes, the attribute matrix is substituted with the identity matrix, allowing the GCN to operate in a semi-supervised manner. This approach effectively enables community detection without requiring fully labeled datasets.

ComE+ [8] is a state-of-the-art community detection algorithm that combines embedding techniques with modularity optimization. It first generates node embeddings that capture the structural features of the network. These embeddings are then used to initialize community assignments, which are iteratively refined to maximize modularity. CommE+ aims to balance the trade-off between computational efficiency and detection accuracy, providing robust performance across various types of networks.

Our experimental results demonstrate that our solution consistently outperforms the Louvain algorithm, GCN, and CommE+ in terms of both NMI and modularity scores across multiple benchmark datasets.

The performance comparison of these algorithms is illustrated in Figures.

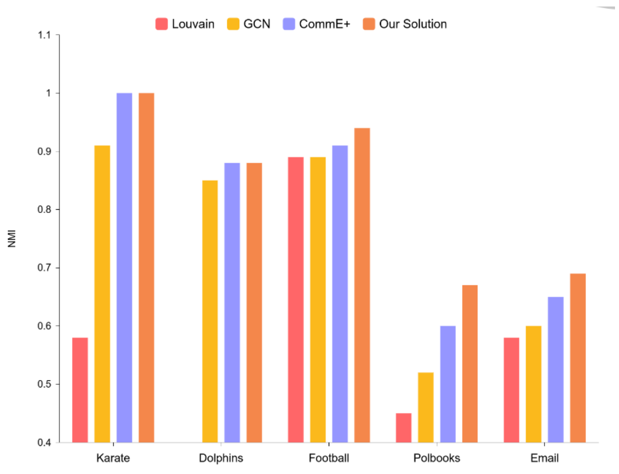

Fig. 10: NMI comparison for different algorithms across real datasets.

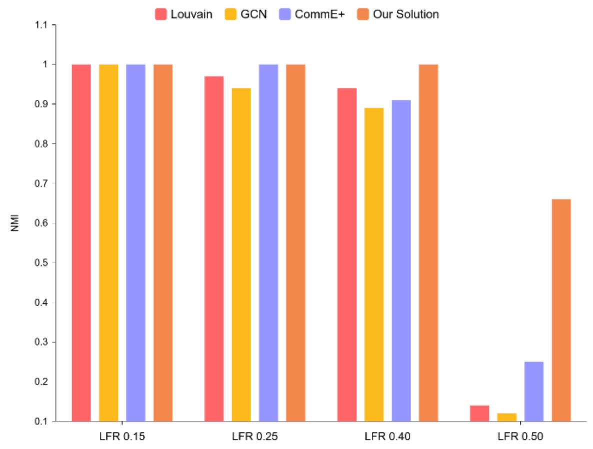

Fig. 11: NMI comparison for different algorithms across synthetic datasets.

5. CONCLUSION

In this paper, we introduced an innovative methodology that combines graph embeddings with clustering techniques in hyperbolic spaces to detect communities. Our solution involves adding a novel clustering technique, X-medoids, to the output of the graph embedding process. This combination utilizes first-, second-, and third-order proximity measures to embed the graph in hyperbolic space and applies X-medoids clustering within the same hyperbolic framework.

We rigorously evaluated our method on a variety of benchmark datasets, encompassing a range of sizes and community structures. The experimental results demonstrated the superiority of our approach compared to several competing configurations, particularly in terms of NMI and modularity. Notably, our solution exhibited a remarkable ability to accurately and robustly detect community structures in diverse networks, such as theemail network and the football network.

Looking ahead, our research could benefit from incorporating infinite Gaussian mixture models (IGMM) to enhance our analytical framework. IGMM offers superior flexibility by automatically estimating the number of communities K, which is crucial for effectively handling networks with complex and potentially infinite community structures. By integrating IGMM into our approach, we can further explore community nuances in hyperbolic space, thereby improving the accuracy and scalability of our method.

We hope that our contributions will significantly advance the field of community detection in complex networks. By opening the door to exploring community detection in hyperbolic spaces, our work provides a new perspective and paves the way for future research in network analysis beyond traditional Euclidean frameworks.

REFERENCES

[1] Political books network. KONECT - The Koblenz Network Collection.

[2] Bijan Afsari. Riemannian l^p center of mass: existence, uniqueness, and convexity. *Proceedings of the American Mathematical Society*, 139(2):655–673, 2011.

[3] Marc Arnaudon, Frédéric Barbaresco, and Le Yang. Riemannian medians and means with applications to radar signal processing. *IEEE Journal of Selected Topics in Signal Processing*, 7(4):595–604, 2013.

[4] Marc Arnaudon, Clément Dombry, Anthony Phan, and Le Yang. Stochastic algorithms for computing means of probability measures. *Stochastic Processes and their Applications*, 122(4):1437–1455, 2012.

[5] Marc Arnaudon and Laurent Miclo. Means in complete manifolds: uniqueness and approximation. *ESAIM: Probability and Statistics*, 18:185–206, 2014.

[6] Marc Arnaudon, Le Yang, and Frédéric Barbaresco. Stochastic algorithms for computing p-means of probability measures, geometry of radar toeplitz covariance matrices and applications to hr doppler processing. In *2011 12th International Radar Symposium (IRS)*, pages 651–656. IEEE, 2011.

[7] Silvere Bonnabel. Stochastic gradient descent on riemannian manifolds. *IEEE Transactions on Automatic Control*, 58(9):2217–2229, 2013.

[8] Sandro Cavallari, Erik Cambria, Hongyun Cai, Kevin Chen-Chuan Chang, and Vincent W. Zheng. Embedding both finite and infinite communities on graphs, August 2019.

[9] Sandro Cavallari, Vincent W Zheng, Hongyun Cai, Kevin Chen-Chuan Chang, and Erik 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.

[10] Tanmoy Chakraborty, Ayushi Dalmia, Animesh Mukherjee, and Niloy Ganguly. Metrics for community analysis: A survey. *arXiv preprint arXiv:1410.6074*, 2014.

[11] Ines Chami. *Representation Learning and Algorithms in Hyperbolic Spaces*. PhD thesis, Stanford University, June 2021.

[12] Aaron Clauset, M. E. J. Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical Review E*, 70(6):066111, 2004.

[13] A. Dhumal and P. Kamde. Survey on community detection in online social networks. *International Journal of Computer Applications*, 121(9), 2015. Available online at www.ijcaonline.org.

[14] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.

[15] J. Friedman, T. Hastie, and R. Tibshirani. *The Elements of Statistical Learning*, volume 1 of *Springer Series in Statistics*. Springer, Berlin, 2001.

[16] Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. *Advances in neural information processing systems*, 31, 2018.

[17] Thomas Gerald, Hadi Zaatiti, Hatem Hajri, Nicolas Baskiotis, and Olivier Schwander. A hyperbolic approach for learning communities on graphs. *Data Mining and Knowledge Discovery*, 37:1090–1124, 2023. HAL Id: hal-04022426, Submitted on 28 Feb 2024.

[18] Sigurdur Helgason. *Differential geometry, Lie groups, and symmetric spaces*. Academic press, 1979.

[19] M. A. Javed, M. S. Younis, S. Latif, J. Qadir, and A. Baig. Community detection in networks: A multidisciplinary review. *Journal of Network and Computer Applications*, 108:87–111, 2018.

[20] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[21] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E*, 78(4):046110, Oct 2008.

[22] Yafang Li, Caiyan Jia, and Jian Yu. A parameter-free community detection method based on centrality and dispersion of nodes in complex networks. *Physica A: Statistical Mechanics and its Applications*, 438:321–334, 2015.

[23] David Lusseau. The emergent properties of a dolphin social network. *Behavioral Ecology and Sociobiology*, 54(4):396–405, 2003.

[24] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

[25] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69(6):066133, 2004.

[26] Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. *Advances in neural information processing systems*, 30, 2017.

[27] Bryan Perozzi, Rami Al-Rfou, and Steven 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.

[28] LKPJ Rdusseeun and P Kaufman. Clustering by means of medoids. In *Proceedings of the statistical data analysis based on the L1 norm conference, neuchatel, switzerland*, volume 31, 1987.

[29] Stanford University. Stanford Large Network Dataset Collection. `http://snap.stanford.edu/`.

[30] Statista. Number of worldwide social network users.

[31] Shi-Kai Sui, Jian-Ping Li, Jian-Guo Zhang, and Shi-Jie Sui. The community detection based on svm algorithm. 2016. Chengdu 611731, China; Hangzhou 310018, China; Zhengzhou 450000, China.

[32] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web (WWW)*, pages 1067–1077. International World Wide Web Conferences Steering Committee, 2015.

[33] Abraham Ungar. *A gyrovector space approach to hyperbolic geometry*. Springer Nature, 2022.

[34] Feifan Wang, Baihai Zhang, Senchun Chai, and Yuanqing Xia. An extreme learning machine-based community detection algorithm in complex networks. *Complexity*, 2018:1–10, 2018.

[35] Xiaofeng Wang, Jianhua Li, Li Yang, and Hongmei Mi. Unsupervised learning for community detection in attributed networks based on graph convolutional network. *Neurocomputing*, 456:147–155, 2021.

[36] Shao-Wu Zhang Wei-Feng Guo. A general method of community detection by identifying community centers with affinity propagation. *Physica A*, 447, 2016.

[37] Xiang Zhao, Yantao Li, and Zehui Qu. A density-based clustering model for community detection in complex networks. 1955:040161, 2018. College of Computer and Information Sciences, Southwest University.