Discrete Ricci Flow applied to Complex Networks

Ollivier-Ricci Curvature and Its Relevance to Community Detection

Lorenzo Fabbri
Giancarlo Oancia

This essay is submitted for the exam of
Complex Networks

Department of Physics
Theoretical Physics Curriculum
University of Bologna
01/03/2025

ABSTRACT

This project uses Ricci Flow as a geometric approach to detect community structures in networks. Specifically, it applies Ollivier-Ricci curvature to adjust edge weights in a graph, iterating the process to shrink intra-community edges and stretch inter-community edges. Then, surgery is performed to separate the graph into distinct connected components, representing communities.

After having tested the Ricci Flow method on synthetic graphs, we applied it to a real dataset: Zachary's Karate Club graph [1], aiming to accurately identify its pre-labelled communities.

Contents

Chapter 1

Overview of the Project

The study of networks has gained considerable attention in various fields, ranging from sociology to biology, and beyond. One of the central problems in network science is community detection, where the objective is to identify groups of nodes (communities) that are more densely connected internally than with the rest of the network. Traditional methods for community detection often rely on statistical or combinatorial approaches. However, recent developments in geometric methods have introduced new approaches to this problem by leveraging concepts from differential geometry [2].

A powerful geometric tool, the Ricci Flow, originally developed in the context of smooth Riemannian manifolds, can be adapted to discrete network structures. In its original formulation, the Ricci Flow evolves the metric of a manifold according to the curvature (represented by the Ricci tensor), leading to a smoothing process over time. Ollivier-Ricci curvature, a discrete approximation of Ricci curvature for graphs, provides a framework to extend this idea to networks, where the "curvature" of edges encodes structural information about node connectivity. Specifically, positive curvature tends to shrink intra-community edges, while negative curvature expands inter-community edges [2].

In this project, we apply Ollivier-Ricci curvature and Ricci Flow to detect the two known communities in Zachary's Karate Club graph. The approach follows the work of Ni et al. [2], where Ricci Flow is used to reshape edge weights iteratively, enhancing the separation between different communities. After applying Ricci Flow, we perform edge surgery to remove weakly connected edges and extract communities as the connected components of the resulting graph.

The results will be compared to the predefined community labels, allowing for a direct comparison between the detected communities and the actual community structures.

The developed code can be accessed in the corresponding GitHub repository: *RicciFlowNetwork*. Code documentation is accessible at *CodeDocumentation*.

To conclude this introductory chapter, we provide here a concise map of how this document is organized and how each chapter contributes to the overall discussion.

Given the need for a rigorous framework, chapter 2 opens with an overview of differentiable manifolds, needed for the formal definition of Ricci Flow. Then, we move to the formal definition of Ricci Flow itself, shifting from smooth differential settings to the discrete analog known as Ollivier-Ricci curvature. This transition is relevant for the application of Ricci Flow in a network-oriented context.

Chapter 3 analyses the historical motivations behind the Ricci Flow, viewing it as a geometrical tool to tackle topological problems. In particular, the Poincaré Conjecture is introduced as an example of such a problem, laying the foundations for why Ricci Flow became central in resolving deep topological questions. To complete the picture, a brief detour into string theory and non-linear sigma models highlights how Poincaré's Conjecture was eventually proved through Perelman's work, showing an instance of Ricci Flow in action.

In Chapter 4, we shift focus to the application to complex networks. Exploiting curvature as a measure of structural cohesion, the chapter illustrates how Ricci Flow techniques can lead to a community identification through separation of the graph into connected components. Furthermore, we discuss the relevance of ARI and modularity within this framework, and compare different methods for Ollivier-Ricci curvature detection.

Finally, chapter 5 is devoted to the code that implements these ideas. It analyses our code and the results obtained for two synthetic graphs (for testing purposes) and for a real dataset (i.e., the Karate Club graph).

CHAPTER 2

DIFFERENTIAL GEOMETRY AND RICCI FLOW

Initially introduced by Richard S. Hamilton in the early 1980s, Ricci Flow emerged as a powerful method in differential geometry. At its core, it seeks to "smooth out" geometric irregularities of manifolds by evolving the underlying Riemannian metric through a partial differential equation similar to the classical heat equation. Recall that a manifold is a topological space that locally resembles a Euclidean space, and a *Riemannian manifold* is such a space equipped with an inner product on each tangent space, allowing for measurements of angles, distances, and curvature.

Curvature, in particular, is fundamental to geometry, as it describes how space bends or deviates from flatness. For example, on a two-dimensional surface embedded in a three-dimensional space, curvature can be visualized by examining the deviation of geodesics (the generalization of "straight lines" in curved spaces) from parallelism, or by looking at how areas and angles are distorted compared to those in flat Euclidean geometry. Extending to higher dimensions and more abstract manifolds requires careful definitions but retains this key notion of "spatial bending."

2.1 Basics of Differential Geometry

To understand the applications of Ricci Flow, we need to explore the properties of Riemannian manifolds, particularly how to define a flow on a manifold and its relation to differential equations. We will see that the Ricci Flow can be viewed as a differential equation for a family of metrics on a Riemannian manifold.

2.1.1 Differentiable Manifolds

Let's begin by defining a *differentiable manifold*, [3, 4].

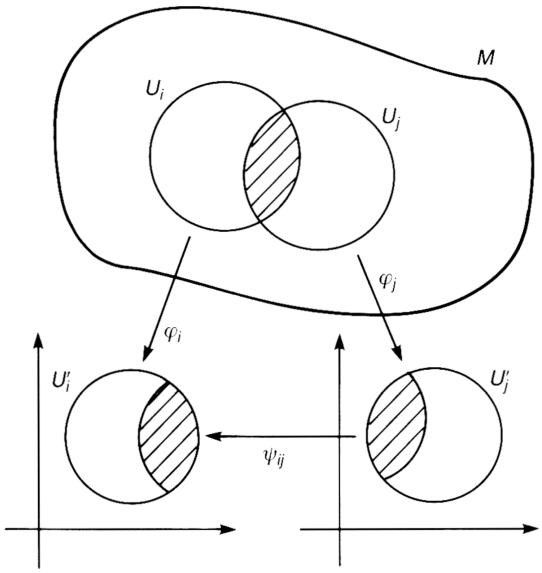


Figure 2.1 φ_i is a homeomorphism from U_i onto an open subset U_i' of $\mathbb{R}^n$.

Definition 1 (Differentiable manifold)**.** An n-dimensional *differentiable manifold* $\mathcal{M}$ is a topological space equipped with a family of pairs $\{(U_i, \varphi_i)\}$, called an *atlas*, where

- Each U_i is an open set in $\mathcal{M}$, and $\cup_i U_i = \mathcal{M}$.
- Each φ_i is a homeomorphism from U_i onto an open subset $U_i' \subseteq \mathbb{R}^n$, as shown in fig. 2.1
- For any U_i and U_j with $U_i \cap U_j \neq \emptyset$, the map

$$\varphi_i \circ \varphi_j^{-1} \colon \varphi_j(U_i \cap U_j) \to \varphi_i(U_i \cap U_j),$$

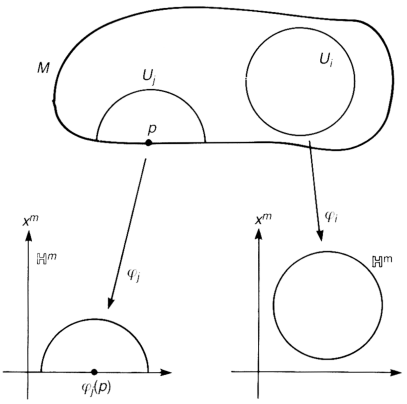
  is infinitely differentiable.

Each pair (U_i, φ_i) is called a *chart*, with U_i the *coordinate neighbourhood* and φ_i the *coordinate map*. Basically, φ_i assigns n real *coordinates* $\{x_1(p), \ldots, x_n(p)\}$ to each point of U_i.

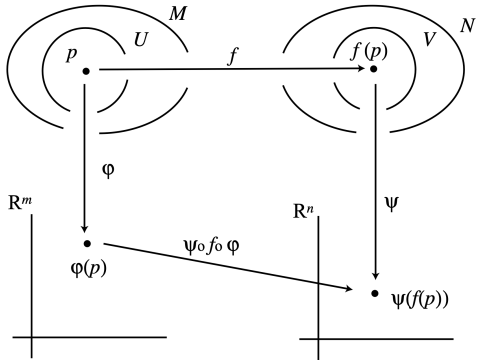
A *manifold with boundary* is defined similarly, except each chart maps into the closed half-space $H^n = \{(x^1, \ldots, x^n) \in R^n | x^n \geq 0\}$, as showed in fig. 2.2a.

2.1.2 Differentiable Maps

Because each chart maps locally to $\mathbb{R}^n$, we adopt the usual notion of differentiability. Let $\mathcal{M}$ and $\mathcal{N}$ be manifolds of dimension m and n, respectively. Consider a map



(a) A manifold $\mathcal{M}$ with boundary. Here, $p \in \partial\mathcal{M}$.



(b) A map $f\colon \mathcal{M} \to \mathcal{N}$ has coordinates representation $\psi \circ f \circ \varphi^{-1}\colon \mathbb{R}^m \to \mathbb{R}^n$.

Figure 2.2 Manifolds with boundaries and maps.

$f : \mathcal{M} \to \mathcal{N}$, as shown in fig. 2.2b. Considering the charts (U, φ) on $\mathcal{M}$ and (V, ψ) on $\mathcal{N}$, the coordinate representation of f is

$$\varphi \circ f \circ \varphi^{-1} : \mathbb{R}^m \to \mathbb{R}^n. \tag{2.1}$$

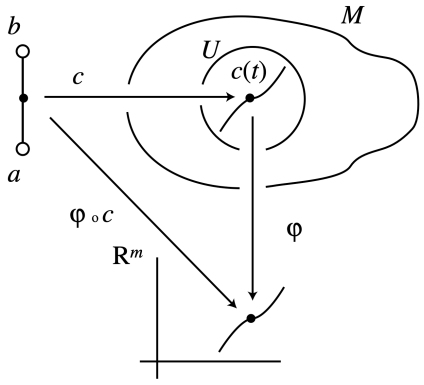
Relaxing the notation, we may write $\varphi(p) = \{x^\mu\}$ and $\psi(f(p)) = \{y^\alpha\}$, so that f is *differentiable* at $p \in \mathcal{M}$ if $y^\alpha = f^\alpha(x^\mu)$ is.

Three special classes of differentiable maps are especially relevant.

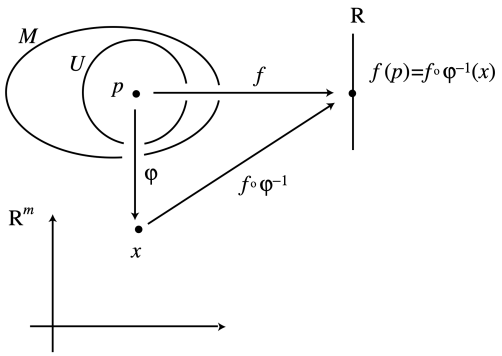
Definition 2 (Diffeomorphism)**.** Let $f\colon \mathcal{M} \to \mathcal{N}$ be a homeomorphism and ψ and φ the same coordinate functions as before. Then, if $\psi \circ f \circ \varphi^{-1}$ is invertible and both $y \equiv \psi \circ f \circ \varphi^{-1}(x)$ and $x \equiv \varphi \circ f^{-1} \circ \psi^{-1}(y)$ are C^∞, f is called a *diffeomorphism* and $\mathcal{M}$ is said to be *diffeomorphic* to $\mathcal{N}$, $\mathcal{M} \equiv \mathcal{N}$.

Definition 3 (Curve)**.** An *open curve* in an n-dimensional manifold $\mathcal{M}$ is a map $c\colon (a, b) \to \mathcal{M}$, where (a, b) is an open interval such that $a < 0 < b$. A *closed curve* is a map $c\colon S^1 \to \mathcal{M}$. On a chart (U, φ), a curve $c(t)$ has the coordinate representation $x = \varphi \circ c\colon \mathbb{R} \to \mathbb{R}^n$. See fig. 2.3a

Definition 4 (Function)**.** A *function* f on $\mathcal{M}$ is a smooth map from $\mathcal{M}$ to $\mathbb{R}$. On a chart (U, φ), the coordinate representation of f is given by $f \circ \varphi^{-1}\colon \mathbb{R}^n \to \mathbb{R}$, which is a real-valued function of n variables. The set of functions is denoted by $\mathfrak{F}(\mathcal{M})$. See fig. 2.3b



(a) A curve c in $\mathcal{M}$ and its coordinate representation $\psi \circ c$.



(b) A function $f\colon \mathcal{M} \to \mathbb{R}$ and its coordinate representation $f \circ \varphi^{-1}$.

Figure 2.3 Curves and functions on a manifold.

2.1.3 Vectors

A useful way to define a *vector* on a manifold is through directional derivatives of functions along curves. Let $c\colon (a,b) \to \mathcal{M}$ be a curve with $c(0) = p$, and let $f\colon \mathcal{M} \to \mathbb{R}$ be any smooth function, as showed in fig. 2.4. The *tangent vector* at p is the directional derivative of $f(c(t))$ along the curve $c(t)$ at $t = 0$, that is,

$$X[f] := \left.\frac{\mathrm{d}f(c(t))}{\mathrm{d}t}\right|_{t=0} = \frac{\partial f}{\partial x^\mu}\left.\frac{\mathrm{d}x^\mu(c(t))}{\mathrm{d}t}\right|_{t=0} = X^\mu\left(\frac{\partial f}{\partial x^\mu}\right), \tag{2.2}$$

where we defined

$$X = X^\mu\left(\frac{\partial}{\partial x^\mu}\right), \quad X^\mu = \left.\frac{\mathrm{d}x^\mu(c(t))}{\mathrm{d}t}\right|_{t=0}. \tag{2.3}$$

Two curves $c_1(t)$ and $c_2(t)$ define the *same* tangent vector at p if

$$c_1(0) = c_2(0) = p \tag{2.4}$$

and

$$\left.\frac{\mathrm{d}x^\mu(c_1(t))}{\mathrm{d}t}\right|_{t=0} = \left.\frac{\mathrm{d}x^\mu(c_2(t))}{\mathrm{d}t}\right|_{t=0}. \tag{2.5}$$

The above relation defines an *equivalence class*, and a vector X at $p \in \mathcal{M}$ is an equivalence class of curves, that is,

$$[c(t)] = \left\{\tilde{c}(t) \;\middle|\; \tilde{c}(0) = c(0) \text{ and } \left.\frac{\mathrm{d}x^\mu(\tilde{c}(t))}{\mathrm{d}t}\right|_{t=0} = \left.\frac{\mathrm{d}x^\mu(c(t))}{\mathrm{d}t}\right|_{t=0}\right\}. \tag{2.6}$$

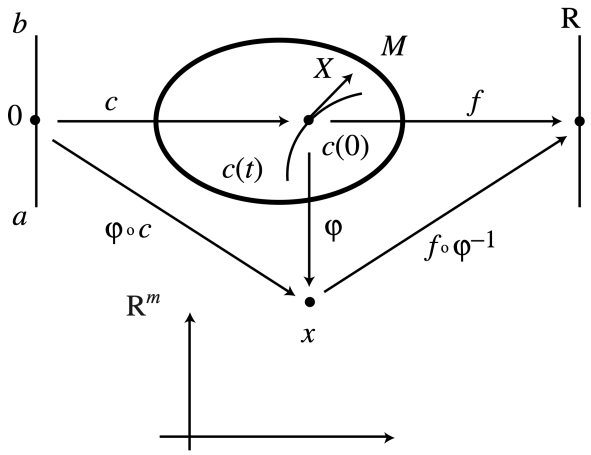


Figure 2.4 A curve c and a function f define a tangent vector along the curve in terms of directional derivatives.

The space of all such vectors at p is the *tangent space* $T_p\mathcal{M}$. A basis for $T_p\mathcal{M}$ is given by

$$\left\{ e_\mu = \frac{\partial}{\partial x^\mu} \right\}, \quad \mu = 1, \ldots, n, \tag{2.7}$$

making $\dim T_p\mathcal{M} = \dim \mathcal{M} = n$. Further, for each $V \in T_p\mathcal{M}$, we can expand it as $V = V^\mu e_\mu = V^\mu \partial_\mu$, and we call V^μ the components of V with respect to the basis.

2.1.4 One-Forms

Since each $T_p\mathcal{M}$ is a vector space, its *dual* space $T_p^*\mathcal{M}$, called the *cotangent space*, consists of linear functionals on $T_p\mathcal{M}$. An element

$$T_p^*\mathcal{M} \ni \omega \colon T_p\mathcal{M} \to \mathbb{R} \tag{2.8}$$

is called *one-form* at p. The simplest example is the *differential* $\mathrm{d}f$ of a function $f \in \mathfrak{F}(\mathcal{M})$. Its action on a vector $V \in T_p\mathcal{M}$ is given by

$$\langle \mathrm{d}f, V \rangle \equiv V[f] = V^\mu \frac{\partial f}{\partial x^\mu} \in \mathbb{R}. \tag{2.9}$$

In coordinates $x = \varphi(p)$, it can be expanded as

$$\mathrm{d}f = \frac{\partial f}{\partial x^\mu}\, \mathrm{d}x^\mu, \tag{2.10}$$

which clearly shows that $\{\mathrm{d}x^\mu\}$ is a basis of $T_p^*\mathcal{M}$, dual to $\{\partial_\mu\}$, since

$$\langle \mathrm{d}x^\mu, \partial_\mu \rangle = \frac{\partial x^\nu}{\partial x^\mu} = \delta^\nu_\mu. \tag{2.11}$$

For a generic one-form $\omega \in T_p^*\mathcal{M}$ and a generic vector $V \in T_p\mathcal{M}$, the *inner product*

$$\langle ., . \rangle : T_p^*\mathcal{M} \times T_p\mathcal{M} \to \mathbb{R} \tag{2.12}$$

is defined by

$$\langle \omega, V \rangle = \omega_\mu V^\mu \langle \mathrm{d}x^\mu, \partial_\nu \rangle = \omega_\mu V^\nu \delta^\mu_\nu = \omega_\mu V^\mu. \tag{2.13}$$

2.1.5 Tensors and Tensor Fields

In multilinear algebra, a *tensor* of type (q, r) at $p \in \mathcal{M}$ is a multilinear map

$$T^{(q,r)} : \underbrace{T_p^*\mathcal{M} \otimes \cdots \otimes T_p^*\mathcal{M}}_{q \text{ times}} \otimes \underbrace{T_p\mathcal{M} \otimes \cdots \otimes T_p\mathcal{M}}_{r \text{ times}} \longrightarrow \mathbb{R}. \tag{2.14}$$

In coordinates, picking dual basis, such a tensor can be expanded as

$$T^{(q,r)} = T^{\mu_1 \ldots \mu_q}{}_{\nu_1 \ldots \nu_r} \frac{\partial}{\partial x^{\mu_1}} \cdots \frac{\partial}{\partial x^{\mu_q}} \mathrm{d}x^{\nu_1} \ldots \mathrm{d}x^{\nu_r}. \tag{2.15}$$

The set of type (q, r) tensors at $p \in \mathcal{M}$ is denoted by $\mathcal{T}^q_{r;p}(\mathcal{M})$. A vector is a $(1, 0)$ tensor, while a one-form is a $(0, 1)$ tensor.

In order to probe a manifold's structure, it's necessary to introduce fields, which are smooth objects defined for each point of $\mathcal{M}$. Knowing the tensor structure at a point, it's easy to generalize it to the entire manifold.

Considering as an example a $(1, 0)$ tensor, a *vector field* is a vector assigned smoothly to each point of $\mathcal{M}$. In other words, V is a vector field if $V[f] \in \mathfrak{F}(\mathcal{M})$, for any $f \in \mathfrak{F}(\mathcal{M})$. Hence, each component of a vector field is itself a smooth function from $\mathcal{M}$ to $\mathbb{R}$. We denote the set of vector fields with $\mathfrak{X}(\mathcal{M})$. Then, a vector $X \in \mathfrak{X}(\mathcal{M})$ computed at a point $p \in \mathcal{M}$ is a vector at $T_p\mathcal{M}$, meaning $X|_p \in T_p\mathcal{M}$.

Taking two vector fields $X, Y \in \mathfrak{X}(\mathcal{M})$, the *Lie bracket*, or *commutator*, $[X, Y]$ is another vector field, which acts on a generic function $f \in \mathfrak{F}(\mathcal{M})$ by

$$[X, Y](f) = X(Y[f]) - Y(X[f]). \tag{2.16}$$

Generalizing, a *tensor field* of type (q, r) is a smooth assignment of an element of $\mathcal{T}^q_{r;p}(\mathcal{M})$ at each point $p \in \mathcal{M}$. The set of tensor fields of type (q, r) on $\mathcal{M}$ is denoted by $\mathcal{T}^q_r(\mathcal{M})$.

2.1.6 Flow generated by a vector field

A smooth vector field on a manifold naturally gives rise to a flow, a continuous mapping that describes how points move along the trajectories defined by the field.

Specifically, if X is a vector field on $\mathcal{M}$, its associated flow is given by the map

$$\sigma \colon \mathbb{R} \times \mathcal{M} \to \mathcal{M}, \tag{2.17}$$

such that for each point x in $\mathcal{M}$, the curve defined by $t \mapsto \sigma(t, x)$ is an integral curve of X. In other words, at each parameter value t the tangent vector to the curve coincides with the value of X at the point $\sigma(t, x)$.

In local coordinates, if we write X in the form

$$X = X^\mu \partial_\mu, \tag{2.18}$$

then the integral curve passing through an initial point x_0 satisfies the system of ordinary differential equations

$$\frac{\mathrm{d}}{\mathrm{d}t}\sigma^\mu(t, x_0) = X^\mu(\sigma(t, x_0)), \tag{2.19}$$

with the initial condition

$$\sigma^\mu(0, x_0) = x_0^\mu. \tag{2.20}$$

A key property of the flow is its group-like behaviour. Indeed, for any two real numbers t and s and for any $x \in \mathcal{M}$, we have

$$\sigma(t, \sigma^\mu(s, x)) = \sigma(t + s, x), \tag{2.21}$$

This property states that flowing first for a parameter s and then for an additional parameter t is equivalent to flowing continuously for a parameter $t + s$. It also implies that each mapping $\sigma(t, \cdot)$ is invertible with inverse $\sigma(-t, \cdot)$.

Applying the existence and uniqueness theorems for ordinary differential equations, one can prove the following theorem.

Theorem T.1 (Fundamental Existence Theorem for Flows). *For any point $x \in \mathcal{M}$, there exists a unique differentiable map $\sigma : \mathbb{R} \times \mathcal{M} \to \mathcal{M}$, satisfying*

- $\sigma(0, x) = x$;
- $t \mapsto \sigma(t, x)$ *is a solution of* (2.19) *and* (2.20);
- $\sigma(t, \sigma^{\mu}(s, x)) = \sigma(t + s, x)$.

It is important to note that if the vector field X is complete — that is, if every integral curve can be extended for all values of the parameter — then the flow is globally defined on $\mathbb{R}$. In contrast, if X is not complete, the flow might only exist around $t = 0$.

Just as flows generated by vector fields encode the dynamics of points on a manifold, the Ricci flow encodes the information about the evolution of a manifold's metric. The correct setting to work in is a Riemannian manifold.

2.1.7 Riemannian Manifolds

A Riemannian manifold is a smooth manifold equipped with a Riemannian metric [5].

Definition 5. Let $\mathcal{M}$ be a differentiable manifold. A *Riemannian metric* g on $\mathcal{M}$ is a type $(0, 2)$ tensor field on $\mathcal{M}$ such that, at each point $p \in \mathcal{M}$:

- $g_p(U, V) = g_p(V, U)$,
- $g_p(U, V) \geq 0$, where the equality holds only when $U = 0$.

Here, $U, V \in T_p\mathcal{M}$ and $g_p = g|_p$. Basically, g_p is a symmetric positive-definite bilinear form.

Recall the previous definition of inner product (2.13) between vectors and dual forms. For $V \in T_p\mathcal{M}$ and $\omega \in T_p^*\mathcal{M}$, the inner product is given by $\langle ., . \rangle : T_p^*\mathcal{M} \times T_p\mathcal{M} \to \mathbb{R}$. If there exists a metric tensor g, then, we can use it to define the inner product between two vectors $U, V \in T_p\mathcal{M}$, specifically by $g_p(U, V)$. Since $g_p : T_p\mathcal{M} \otimes T_p\mathcal{M} \to \mathbb{R}$, we may define a linear map $g_p(U, .) : T_p\mathcal{M} \to \mathbb{R}$ by $V \mapsto g_p(U, V)$. Then, it's straightforward that $g_p(U, .) \in T_p^*\mathcal{M}$ is a one-form. Thus, the metric g_p gives rise to an isomorphism between $T_p\mathcal{M}$ and $T_p^*\mathcal{M}$.

Choosing a chart (φ, U), with coordinates $\{x^\mu\}$, we can express the metric tensor in local coordinates as

$$g_p = g_{\mu\nu}(p)\,\mathrm{d}x^\mu \otimes \mathrm{d}x^\nu, \quad g_{\mu\nu}(p) = g_p(\partial_\mu, \partial_\nu) = g_{\nu\mu}(p), \quad p \in \mathcal{M}. \tag{2.22}$$

It is usual convention to omit the point p, denote the inverse metric as $g^{\mu\nu}$, and the determinant as $\det(g_{\mu\nu}) \coloneqq g$ and $\det(g^{\mu\nu}) \coloneqq g^{-1}$. Thus, the isomorphism between $T_p\mathcal{M}$ and $T_p^*\mathcal{M}$ can be expressed as

$$\omega_\mu = g_{\mu\nu}U^\nu, \quad U^\mu = g^{\mu\nu}\omega_\nu. \tag{2.23}$$

One implicitly refers to this isomorphism by saying that the metric allows raising and lowering indices.

In a Riemannian manifold, the standard notion of differentiation does not preserve the tensorial character of geometric objects. To overcome this difficulty, we introduce the concept of the *covariant derivative*. It extends the idea of directional derivatives to curved spaces, and allows us to define the concept of parallelism on a manifold.

2.1.8 Covariant Derivatives

We first introduce the concept of an *affine connection*.

Definition 6. An *affine connection* ∇ is a map $\nabla\colon \mathfrak{X}(\mathcal{M}) \times \mathfrak{X}(\mathcal{M}) \to \mathfrak{X}(\mathcal{M})$, or $(X, Y) \mapsto \nabla_X Y$ which satisfies the following conditions

$$\nabla_X(Y + Z) = \nabla_X Y + \nabla_X Z \tag{2.24a}$$
$$\nabla_{(X+Y)}Z = \nabla_X Z + \nabla_Y Z \tag{2.24b}$$
$$\nabla_{(fX)}Y = f\nabla_X Y \tag{2.24c}$$
$$\nabla_X(fY) = X[f]Y + f\nabla_X Y, \tag{2.24d}$$

where $f \in \mathfrak{F}(\mathcal{M})$ and $X, Y, Z \in \mathfrak{X}(\mathcal{M})$.

Take a chart (U, φ) with the coordinate $x = \varphi(p)$ on $\mathcal{M}$, and define the functions $\Gamma^\lambda_{\nu\mu}$, called *Christoffel symbols*, by

$$\nabla_\nu e_\mu \equiv \nabla_{e_\nu} e_\mu = e_\lambda \Gamma^\lambda_{\nu\mu}, \tag{2.25}$$

where $\{e_\mu\} = \{\partial_\mu\}$ is the coordinate basis in $T_p\mathcal{M}$. By computing their transformation rule under a generic change of coordinates, one can show that Christoffel symbols are *not* components of a tensor.

Once the action of ∇ on the basis vectors is defined, we can compute its action on any vector, i.e.,

$$\nabla_V W = V^\mu \left(\frac{\partial W^\lambda}{\partial x^\mu} + W^\nu \Gamma^\lambda_{\mu\nu} \right) e_\lambda. \tag{2.26}$$

In particular, ∇ maps two vectors V and W to a new vector given by the right-hand side of eq. (2.26), whose λth component is $V^\mu \nabla_\mu W^\lambda$, where

$$\nabla_\mu W^\lambda \equiv \frac{\partial W^\lambda}{\partial x^\mu} + \Gamma^\lambda_{\mu\nu} W^\nu \tag{2.27}$$

Beyond vector fields, the concept of the covariant derivative extends naturally to tensor fields of any rank. For a general tensor field T of type (q, r), the covariant derivative ∇T is defined to behave as a tensor of type $(q, r+1)$ and to satisfy the appropriate product rule with respect to the tensor contractions and tensor products.

For Riemannian manifolds endowed with a metric tensor g, the affine connection can be chosen to be *compatible* with the metric, meaning that

$$\nabla_X g = 0, \quad \forall X \in \mathfrak{X}(\mathcal{M}). \tag{2.28}$$

Intuitively, this means that angles between vectors are conserved while parallelly transporting them along each other.

Curvature provides a quantitative measure of how a manifold deviates from being flat. Since Γ is not a tensor, it can't have an intrinsic geometric meaning as a measure of the curvature. To serve this purpose, one introduces the *torsion* and *Riemann tensor*.

2.1.9 Curvature and Ricci Tensor

The *torsion tensor* $T\colon \mathfrak{X}(\mathcal{M}) \otimes \mathfrak{X}(\mathcal{M}) \to \mathfrak{X}(\mathcal{M})$ and the *Riemann tensor* $R\colon \mathfrak{X}(\mathcal{M}) \otimes \mathfrak{X}(\mathcal{M}) \otimes \mathfrak{X}(\mathcal{M}) \to \mathfrak{X}(\mathcal{M})$, also called *curvature tensor*, are defined by

$$T(X, Y) \equiv \nabla_X Y - \nabla_Y X - [X, Y], \tag{2.29}$$

$$R(X, Y, Z) \equiv \nabla_X \nabla_Y Z - \nabla_Y \nabla_X Z - \nabla_{[X,Y]} Z. \tag{2.30}$$

For simplicity, we consider torsionless connections, for which the Christoffel symbols are symmetric, meaning that

$$\Gamma^{\lambda}_{\nu\mu} = \Gamma^{\lambda}_{\mu\nu}. \tag{2.31}$$

Further, for a metric compatible connection, one can derive the following expression

$$\Gamma^{\lambda}_{\mu\nu} = \frac{1}{2} g^{\lambda\rho} \left(\partial_\nu g_{\mu\rho} + \partial_\mu g_{\nu\rho} - \partial_\rho g_{\mu\nu} \right), \tag{2.32}$$

and the Riemann tensor has components

$$R_{\mu\nu\rho\sigma} = \frac{1}{2} \left(\partial^2_{\nu\rho} g_{\mu\sigma} - \partial^2_{\nu\sigma} g_{\mu\rho} + \partial^2_{\mu\sigma} g_{\nu\rho} - \partial^2_{\mu\rho} g_{\nu\sigma} \right). \tag{2.33}$$

Using the general definition (2.30), one can prove the following identities

$$R_{\mu\nu\rho\sigma} = -R_{\nu\mu\rho\sigma} = -R_{\mu\nu\sigma\rho}, \tag{2.34a}$$

$$R_{\mu\nu\rho\sigma} = R_{\rho\sigma\mu\nu}, \tag{2.34b}$$

$$R^{\mu}_{\nu\rho\sigma} + R^{\mu}_{\sigma\nu\rho} + R^{\mu}_{\rho\sigma\nu} = 0. \tag{2.34c}$$

The symmetries allow us to define the symmetric tensor $R_{\mu\nu} = R^{\lambda}_{\mu\lambda\nu}$, called *Ricci tensor*, and the *Ricci scalar* $R = R^{\mu}_{\mu}$.

2.2 Definition of Ricci Flow

In Riemannian geometry, the concept of curvature is related to the previously defined quantities.

Concretely, let (M, g) be a Riemannian manifold, where $\mathcal{M}$ is a smooth manifold and g is the metric tensor. As we've just seen, the curvature Ric is derived as a contraction of the Riemann curvature tensor, itself an operator capturing how much nearby geodesics converge or diverge. Positive Ricci curvature typically implies that geodesics tend to converge, reflecting a "crowded" or positively curved geometry akin to the sphere. Negative Ricci curvature implies geodesics tend to diverge, mirroring a hyperbolic or "saddle-like" structure. Zero Ricci curvature represents Ricci-flat manifolds.

2.2.1 Hamilton's Ricci Flow Equation

Hamilton introduced the *Ricci Flow* as the partial differential equation:

$$\frac{\partial g_{ij}}{\partial t} = -2\, R_{ij}, \tag{2.35}$$

where g_{ij} are the components of the metric tensor g in local coordinates and R_{ij} are the components of the Ricci curvature tensor [6]. Informally, each infinitesimal piece of the manifold changes in time, guided by curvature. Regions of *high positive* Ricci curvature shrink faster, while regions of *negative* Ricci curvature expand. This leads to a *flow* that tends to smooth out the geometric and topological features of $\mathcal{M}$, [7].

One of the most famous applications of Ricci Flow on manifolds is Grigori Perelman's resolution of the Poincaré Conjecture and the more general Geometrization Conjecture for three-dimensional manifolds. Perelman's work introduced the notion of *Ricci Flow with surgery*, a procedure to remove singular regions (places where curvature blows up to infinity) and continue the flow on the remaining parts. In 3D manifolds, these singularities can be visualized as "neck pinches" that effectively separate the manifold into topologically simpler pieces. Perelman showed that by performing a series of well-defined surgeries, one could decompose a three-dimensional manifold into model geometric pieces, completing Hamilton's program toward a proof of the Geometrization Conjecture [8].

2.2.2 From Smooth Settings to Discrete Geometry

While Ricci Flow is classically defined on smooth manifolds, there has been considerable interest in transferring these ideas to *discrete* or combinatorial settings such as polyhedral surfaces, graphs, and complex networks. The general question is how to define concepts like "curvature" when one does not have a smooth manifold or a Riemannian metric in the usual sense. Instead, discrete analogs focus on adjacency, distances along edges, and combinatorial properties that mimic or reflect continuum notions.

For surfaces composed of polygons (triangulations), one can define the curvature at a vertex via angle deficits, a concept dating back to classical differential geometry of polyhedral surfaces. However, for higher-dimensional graphs and networks that are not neatly embedded in any Euclidean space, a more general curvature definition is needed—one that depends mostly on the underlying distances and probability measures rather than an explicit embedding.

2.3 Optimal Transport and Ollivier's Ricci Curvature

A major breakthrough in defining a curvature notion for general metric spaces (including discrete networks) came via *optimal transport*. Historically, the optimal transport problem, originating in the work of Gaspard Monge in the 18th century, asks how to map one mass distribution into another with minimal transportation cost. Subsequent reformulations by Leonid Kantorovich turned this into a linear optimization problem known as the *Kantorovich relaxation*.

If (X, d) is a metric space, and we have two probability measures μ and ν on X, the *Wasserstein distance* (also called the earth mover's distance) measures how much "effort" is needed to move mass from μ to ν, given the metric d. Specifically, one solves an optimization problem that tries to minimize the total cost of moving infinitesimal amounts of mass from one location to another.

Yann Ollivier harnessed this framework to define a notion of *coarse Ricci curvature* on general metric measure spaces. Ollivier's definition, now widely called *Ollivier Ricci curvature*, is based on considering small probability balls of radius ε around points (or sometimes discrete probability measures concentrated on nearest neighbours in a graph) and calculating how much these small balls cost to move one onto the other under optimal transport. If moving these balls requires comparatively more effort than just their pairwise distance would suggest, the edge or connection between them is deemed negatively curved; if it requires less effort, it is positively curved. This emerges from an analog of the well-known statement in Riemannian geometry that Ricci curvature controls how geodesic balls deviate from each other, which can be recast in terms of mass transport.

CHAPTER 3

APPLICATIONS TO TOPOLOGY

Before applying Ricci Flow in a complex network context, let us briefly turn to string theory and topology to appreciate its historical and mathematical significance. As discussed in the previous chapter, Ricci Flow was instrumental in proving the Poincaré Conjecture, which we introduce shortly. Moreover, the string-theoretic sigma model provides a concrete instance of Ricci Flow, and we will analyse this next.

3.1 Poincaré Conjecture

To understand the origins of Ricci Flow and its relevance to mathematics, we first need to examine how it applies to geometrical and topological problems. Hence, we begin by introducing the Poincaré conjecture and Perelman's proof.

In his 1895 paper *Analysis Situs* [9], Poincaré laid the foundations for what we now call *topology*. He clearly underlined his purpose in this article, stating:

> *...geometry is the art of reasoning well from badly drawn figures; however, these figures, if they are not to deceive us, must satisfy certain conditions; the proportions may be grossly altered, but the relative positions of the different parts must not be upset.*

Because we don't need to trace every historical development, we now provide modern definitions and intuitions about topology [10], beginning with a formal definition of a topological space.

Definition 7 (Topological space)**.** Let X be a non-empty set. A *topology* on X is a family $\mathcal{A}$ of subsets $\mathcal{A} \ni A \subseteq X$, called *open sets*, such that

- The empty set and X are open sets,

$$\emptyset \in \mathcal{A}, \quad X \in \mathcal{A}. \tag{3.1}$$

- The union of any collection of open sets is again an open set,

$$A_i \in \mathcal{A}, i \in J \implies \bigcup_{i \in J} A_i \in \mathcal{A}, \tag{3.2}$$

  with J a collection of indices.

- The intersection of any finite number of open sets is again an open set,

$$A_i \in \mathcal{A}, i = 1, 2 \ldots, N \implies \bigcap_{i=1}^{N} A_i \in \mathcal{A}. \tag{3.3}$$

The set X, with the topology $\mathcal{A}$, is called a *topological space* $(X, \mathcal{A})$.

We can then define a *closed set* simply by taking the complement of any open set:

Definition 8 (Closed sets)**.** Given $A \in \mathcal{A}$, a subset $C \subseteq X$ is *closed* if

$$C \text{ closed} \iff X \setminus C \text{ open}. \tag{3.4}$$

One crucial property of topological spaces is *compactness*, which captures the idea of finiteness in this context:

Definition 9 (Compact)**.** A topological space $(X, \mathcal{A})$ is called *compact* if every open cover,

$$X \in \bigcup_{\alpha \in C} U_\alpha, \tag{3.5}$$

contains a finite subcover,

$$X \in \bigcup_{i=1}^{N} U_i. \tag{3.6}$$

Another important property is *connectedness*: intuitively, a connected space can't be partitioned into separate pieces with nothing in common. To make this intuition more precise, we first need to define a *path*.

Definition 10 (Path)**.** Let $(X, \mathcal{A})$ be a topological space, and let $x, y \in X$. A *path* is a continuous function $f : [0, 1] \to X$ with $f(0) = x$ and $f(1) = y$. A *path-component* of X is an equivalence class of X where x and y belong to the same class if a path connects them.

Definition 11 (Connected)**.** A topological space $(X, \mathcal{A})$ is *connected* if it can't be represented as the union of two or more disjoint non-empty open subsets.

Definition 12 (Path connected)**.** A topological space $(X, \mathcal{A})$ is *path connected* if there is exactly one path-component. For non-empty spaces, this is equivalent to saying that any two points in X can be joined by a path.

Definition 13 (Simply connected)**.** A topological space $(X, \mathcal{A})$ is *simply connected* if it is path connected and every path between two points can be continuously transformed into any other path connecting the same two points, while keeping the endpoints fixed.

Beyond these properties, we often define continuous maps from one topological space to another. A *homeomorphism* is a continuous, bijective map whose inverse is also continuous, making the two spaces essentially "the same" from a topological viewpoint.

A crucial closed manifold in the definition of the Poincaré conjecture is the 3-sphere.

Definition 14 (3-sphere)**.** In coordinates, a 3-sphere centred at (C_0, C_1, C_2, C_3) with radius r is the set of points $(x_0, x_1, x_2, x_3) \in \mathbb{R}^4$ satisfying

$$\sum_{i=0}^{3}(x_i - C_i)^2 = (x_0 - C_0)^2 + (x_1 - C_1)^2 + (x_2 - C_2)^2 + (x_3 - C_3)^2 = r^2. \tag{3.7}$$

As a manifold, the 3-sphere is a compact, connected 3-dimensional manifold, with no boundary. Poincaré's conjecture can thus be stated [9]:

> *A simply-connected closed manifold is homeomorphic to a sphere.*

Initially, Poincaré did not consider this a conjecture because it seemed trivial, yet it became one of the greatest unsolved problems in mathematics. Refined over time, its final form [11] is:

> *Every three-dimensional topological manifold which is closed, connected, and has trivial fundamental group is homeomorphic to the three-dimensional sphere.*

To avoid going into all the complexities of homology and fundamental groups, we cite a useful result:

Theorem T.2. *A path-connected topological space is simply connected if and only if its fundamental group is trivial.*

Understanding how to prove the Poincaré conjecture requires investigating specific manifold structures, ultimately leading to the application of Ricci Flow.

3.2 Non-Linear Sigma Models and String Theory

The standard starting point of string theory is *Polyakov action*, which represents a bosonic, classical, one-dimensional, string [12, 13]. This latter traces out a two-dimensional worldsheet Σ, whose intrinsic coordinates are $\xi^a = (\tau, \sigma)$, $a = 1, 2$. Further, the string is embedded in a 26-dimensional spacetime, and from this point of view it's described by the bosonic coordinates $X^\mu(\xi)$, $\mu = 0, \dots 25$. We focus on the *closed string*, defined by the periodicity condition $X^\mu(\tau, \sigma) = X^\mu(\tau, \sigma + l)$.

The metric on spacetime is denoted by $g_{\mu\nu}$, while the metric on the worldsheet is γ_{ab}. For a flat spacetime, with Minkowski metric $g_{\mu\nu} \equiv \eta_{\mu\nu} = \text{diag}(-1, +1, \dots, +1)$, the action reads

$$S_P[X^\mu(\xi), \gamma_{ab}(\xi)] = -\frac{1}{4\pi\alpha'} \int_\Sigma \mathrm{d}\tau\, \mathrm{d}\sigma \sqrt{-\det(\gamma)}\, \gamma^{ab} \partial_a X_\mu(\xi) \partial_b X^\mu(\xi), \tag{3.8}$$

where α' is a characteristic parameter of the string, related to the string tension by $T = 1/2\pi\alpha'$.

The symmetries of this action allow us to fix the gauge such that the worldsheet metric is flat, $\gamma_{ab} = \eta_{ab}$. Reintroducing the spacetime metric $g_{\mu\nu}$ for completeness, we obtain

$$S_P = -T \int_\Sigma \mathrm{d}^2\xi\, g_{\mu\nu}(X) \partial_a X^\mu \partial^a X^\nu. \tag{3.9}$$

Basically, this action represents a 2-dimensional field theory on the worldsheet, where the coordinates X^μ are 26 dynamical fields. This allows us to quantize the theory with the usual quantization prescription, based on the substitution of the classical Poisson brackets defined on a symplectic manifold with the commutators of operators acting on a Hilbert space.

After quantization, the massless spectrum of the closed string contains a *graviton* $\gamma_{\mu\nu}$, which resembles a gravitational wave at low energies, a scalar field φ called *dilaton* and an antisymmetric two-tensor $b_{\mu\nu}$ called *Kalb-Ramond tensor*.

Because of the presence of the graviton, one could wonder what happens for a non-flat spacetime. To understand it, it's first necessary to redefine the coordinates as a constant X_0^μ plus some other arbitrary fields Y^μ,

$$X^\mu(\xi) = X_0^\mu(\xi) + \sqrt{\alpha'}\, Y^\mu(\xi). \tag{3.10}$$

The term in the Lagrangian can be expanded as

$$\begin{aligned} & g_{\mu\nu}(X)\partial_a X^\mu \partial^a X^\nu \\ & = \alpha' \Big[g_{\mu\nu}(X_0) + \sqrt{\alpha'} g_{\mu\nu,\rho}(X_0) Y^\rho(\xi) \\ & + \frac{\alpha'}{2} g_{\mu\nu,\rho\sigma}(X_0) Y^\rho(\xi Y^\sigma(\xi)) + \dots \Big] \partial_a Y^\mu \partial^a Y^\nu, \end{aligned} \tag{3.11}$$

where $g_{\mu\nu,\rho} \equiv \partial_\rho g_{\mu\nu}$. We obtained an expansion in α', where each term is an interaction term for the fields Y^μ, with couplings given by the derivatives of the metric.

One additional feature of the Polyakov action (3.8) is the invariance under *conformal transformations*. Those are diffeomorphisms on a Riemannian manifold which preserve the metric up to rescaling, that is,

$$g(x) \to \tilde{g}(\tilde{x}) = e^{2\omega(\tilde{x})} g(\tilde{x}). \tag{3.12}$$

Without going into the details, the presence of this symmetry is considered as a consistency condition for the theory, as it allows for a perturbative interpretation of the interactions.

Since the action which contains (3.11) describes an interacting quantum field theory, it must undergo renormalization, to cure divergences. A generic property of renormalized theories is that the coupling constant is running. This is, however, incompatible with the conformal symmetry (3.12), which implies scale-invariance.

Therefore, to tackle the quantum anomaly of the conformal symmetry, one should impose that the β-*function* vanishes,

$$\beta(g_{\mu\nu}) = M \frac{\partial}{\partial M} g_{\mu\nu} \stackrel{!}{=} 0. \tag{3.13}$$

A similar argument can be pursued for the other two particles in the closed string spectrum, the dilaton φ and the Kalb-Ramond form $b_{\mu\nu}$. First, the generalization of the action (3.9) which includes those fields is

$$S_\sigma = -\frac{T}{2} \int_\Sigma \mathrm{d}^2\xi \sqrt{-\det(\gamma)} \left[\left(\gamma^{ab} g_{\mu\nu}(X) + i\varepsilon^{ab} b_{\mu\nu}(X) \right) \partial_a X^\mu \partial_b X^\nu + \alpha' \mathcal{R} \varphi(X) \right], \tag{3.14}$$

where ε^{ab} is the 2d Levi-Civita symbol, and $\mathcal{R} = \mathcal{R}(\gamma)$ is the Ricci scalar on the worldsheet.

Defining the field strength $H_{\mu\nu\rho} = \partial_\mu b_{\nu\rho} + \partial_\nu b_{\rho\mu} + \partial_\rho b_{\mu\nu}$, the vanishing of the beta functions reads

$$\beta(g_{\mu\nu}) = \alpha' \left(R_{\mu\nu} - \frac{1}{4} H_{\mu\lambda\rho} H^{\mu\lambda\rho} + 2\nabla_\mu \nabla_\nu \varphi \right) + O(\alpha'^2) \stackrel{!}{=} 0, \tag{3.15a}$$

$$\beta(b_{\mu\nu}) = \alpha' \left(\frac{1}{2} \nabla^\rho H_{\rho\mu\nu} + \nabla^\rho \varphi H_{\rho\mu\nu} \right) + O(\alpha'^2) \stackrel{!}{=} 0, \tag{3.15b}$$

$$\beta(\varphi) = \alpha' \left(\frac{1}{2} \nabla_\mu \varphi \nabla^\mu \varphi - \frac{1}{2} \nabla^2 \varphi - \frac{1}{24} H_{\mu\nu\rho} H^{\mu\nu\rho} \right) + O(\alpha'^2) \stackrel{!}{=} 0. \tag{3.15c}$$

The above equations are constraints for the spacetime fields (g, b, φ), imposed to preserve conformal invariance of the quantum string. However, since those fields should be dynamical on spacetime, those must also be their equations of motion. This leads to the following *low-energy effective action*, which has (3.15) as equations of motion

$$S_{26} = \frac{1}{k_0^2} \int \mathrm{d}^{26} x \sqrt{\det(g)} e^{-2\varphi} \left(\mathcal{R}(g) - \frac{1}{12} H_{\mu\nu\rho} H^{\mu\nu\rho} + 4\nabla_\mu \varphi \nabla^\mu \varphi \right). \tag{3.16}$$

To set this problem to a more general ground, and understand how this model is related to Ricci Flow, let's define more accurately what a σ-model is in field theory. A σ-model is a field theory for a field $\Phi \colon \Sigma \to \mathcal{M}$ that takes values in a manifold $\mathcal{M}$. Traditionally, Σ is the spacetime on which the field theory lives, and $\mathcal{M}$ is called the target space. If the target space carries some linear structure, like a vector space, then the whole physical system is called a linear σ-model. For general manifolds such as generic Riemannian ones, it is then called a non-linear σ-model. What we did above is to study the σ-model spacetime renormalization effects on the target space $\mathcal{M}$, and this is indeed an instance of Ricci Flow [14].

CHAPTER 4

APPLICATION TO COMPLEX NETWORKS

To adapt Ollivier's construction to a graph $G = (V, E)$ where V is the set of nodes and E is the set of edges (possibly with weights), one often assigns to each node x a probability measure m_x. A typical choice is to concentrate the mass uniformly on x's neighbours, possibly with some parameter α to keep a fraction of the mass at x itself. Let $W(m_x, m_y)$ denote the Wasserstein distance between m_x and m_y. The Ollivier Ricci curvature $\kappa(x, y)$ along the edge (x, y) is defined as

$$\kappa(x, y) \;=\; 1 \;-\; \frac{W\big(m_x, m_y\big)}{d(x, y)}, \tag{4.1}$$

where $d(x, y)$ is the usual shortest-path distance (or a weight-based distance) between x and y. Intuitively:

- If many neighbours of x align well with those of y, then $W(m_x, m_y)$ is relatively small compared to $d(x, y)$, giving a larger curvature.
- If the neighbours do not overlap much, $W(m_x, m_y)$ will be comparatively large, implying smaller (or possibly negative) curvature.

This lines up with the idea that a highly "clustered" or cohesive set of nodes—often indicating an underlying community—acts more like a positively curved region in the manifold analogy, whereas edges bridging distant clusters reflect negative curvature. These insights lead to a method for analysing and partitioning networks by focusing on edges with specific curvature characteristics.

4.1 General Workflow and Important Parameters

In the study of complex networks, a fundamental task is to identify densely connected groups of nodes, commonly referred to as *communities*. These communities often correspond to meaningful substructures such as friend groups in social networks, functionally related proteins in biological networks, or topics in citation networks. There are numerous algorithms that aim to extract communities—ranging from graph partitioning heuristics and centrality-based edge removal to statistical and probabilistic methods.

The Ricci Flow-based technique for network community detection focuses on the geometric viewpoint: an edge with significantly negative Ricci curvature may signify a *bridge* between distinct communities, whereas edges with positive curvature are typically nestled within cohesive communities. Iteratively adjusting edge weights according to Ollivier Ricci curvature has the effect of "magnifying" bridging edges and "contracting" internal edges, ultimately making a subsequent threshold-based cut reveal the inherent clusters.

The main steps for using Ricci Flow to detect communities are:

1. *Initialization*: Assign an initial weight to each edge (x, y). Often, one starts with uniform weights or with weights based on an existing property such as adjacency or similarity.

2. *Probability Measures*: Choose how to define the measure m_x at each node x. A popular simple choice is to put uniform weight on all neighbours of x, ensuring $\sum_{v \in \text{neighbours}(x)} m_x(v) = 1$. Other weighting schemes (e.g., discounting more distant neighbours) can also be used.

3. *Curvature Computation*: For each edge (x, y), compute the Ollivier Ricci curvature $\kappa(x, y)$ by solving the discrete optimal transport problem and using

$$\kappa(x, y) = 1 - \frac{W(m_x, m_y)}{d(x, y)}. \tag{4.2}$$

4. *Discrete Ricci Flow Update*: Adjust the edge weight according to

$$w_{xy}^{(i+1)} = w_{xy}^{(i)} \; - \; \eta \, \kappa_{xy}^{(i)} \, d_{xy}^{(i)}. \tag{4.3}$$

Often, one sets $\eta = 1$ and updates all edges simultaneously, then recomputes the shortest path distances $d(\cdot, \cdot)$ for the next iteration. The total number of iterations can be chosen based on convergence criteria or practical heuristics.

5. *Network Surgery*: After a certain number of iterations, examine the distribution of edge weights. Typically, bridging edges (those connecting separate communities) will have grown in length. Choose a threshold T such that edges with $w_{xy} > T$ are considered "cuts," removing them from the graph. The connected components that remain are taken as the identified communities.

6. *Post-Processing*: If a graph has hierarchical communities, additional steps (e.g., repeating the process within subcomponents) might be performed to further subdivide the clusters.

In practice, one needs to choose three important parameters:

1. *Number of Ricci Flow Iterations*: Stopping too soon might not emphasize bridging edges enough to isolate communities; iterating too long might lead to degeneracies (e.g., certain edges become extremely large or extremely small). An empirical strategy is to run a moderate number (e.g., 10–20 iterations) and check if measures like ARI (if a ground truth is available) or modularity saturate.

2. *Cut Threshold for Network Surgery*: Typically chosen by scanning multiple thresholds and evaluating a measure of clustering quality (e.g., modularity) or ARI. The threshold essentially encodes when an edge has grown so large that it is acting as a "neck" in geometric terms and should be removed.

3. *Mass Distributions*: The simplest is uniform distribution on neighbours (sometimes with or without a fraction of mass at the node itself). More sophisticated distributions might downweight distant neighbours, especially in large or weighted networks, thereby changing the computed transportation distance (whether OTD or ATD) and influencing the resulting curvature values.

Then, one needs to validate the method and also make it feasible for cases where ground truth is not available. For these purposes we can make use of *Adjusted Rand Index (ARI)* and *Modularity*.

4.1.1 General Definition of ARI

The *Adjusted Rand Index (ARI)* is a popular external validation measure to compare a discovered clustering with a known ground-truth partition. Suppose you have a set

of n items (in our case, the n nodes of a network). Let $C = \{C_1, \ldots, C_r\}$ be a partition of these items into r clusters found by some method, and let $G = \{G_1, \ldots, G_s\}$ be the ground-truth (or reference) partition into s clusters. The Rand Index (RI) measures the fraction of item pairs that are *consistently* assigned in both partitions (i.e., either assigned together in both or assigned to different clusters in both).

Formally, the Rand Index is given by:

$$\mathrm{RI}(C, G) \;=\; \frac{a + d}{a + b + c + d}, \tag{4.4}$$

where

- a is the number of pairs of items that are in the same cluster in C *and* in the same cluster in G;
- b is the number of pairs of items that are in the same cluster in C but in different clusters in G;
- c is the number of pairs that are in different clusters in C but in the same cluster in G;
- d is the number of pairs that are in different clusters in C and in different clusters in G.

Because $a + d$ counts all the agreements (put together or kept apart) and $b + c$ counts the disagreements, $\mathrm{RI}(C, G)$ is between 0 and 1, with 1 meaning a perfect match of the partitions.

However, the Rand Index does not correct for chance agreement. The *Adjusted Rand Index* refines this by subtracting the expected RI of random partitions and rescaling. One can define:

$$\mathrm{ARI}(C, G) \;=\; \frac{\mathrm{RI}(C, G) \;-\; \mathrm{Expected[RI]}}{\max(\mathrm{RI}) \;-\; \mathrm{Expected[RI]}}, \tag{4.5}$$

which yields a value that ranges from 0 (or negative, depending on definition) up to 1. Here, 1 indicates the clustering C exactly matches the ground-truth G, whereas an ARI near 0 suggests random agreement.

4.1.2 Applying ARI to Ricci Flow-based Community Detection

When applying Ricci Flow on a network, one typically obtains a final partition of the graph by removing edges beyond a certain length. If a ground-truth labelling

exists, we compute the ARI to assess how well these communities align with the reference. If ARI is high, that means the geometric approach successfully captured the intended grouping. By examining the ARI as a function of the threshold, one can also decide the best cutoff for the "network surgery." In practice, one might do a range of threshold values, measure the ARI for each, and pick the threshold that maximizes it (assuming the ground-truth is known). In contexts where the ground-truth partition is not known, we might rely on internal validation measures, such as modularity, to guess a suitable threshold.

4.1.3 Definition of Modularity

Modularity is one of the most commonly used internal metrics for community detection in networks. Proposed initially by Newman and Girvan, it quantifies how well a particular partition of the network divides the nodes into communities that are dense internally and sparse between each other.

Let us consider a network with n nodes and m edges (or total edge weight if it is a weighted graph). For a given partition of the network into k communities, the modularity Q is computed as:

$$Q \;=\; \frac{1}{2m}\sum_{i,j}\Big(A_{ij} \;-\; \frac{d_i\,d_j}{2m}\Big)\,\delta(c_i,c_j), \tag{4.6}$$

where A_{ij} is the adjacency matrix (or the weight matrix), d_i is the degree (or sum of weights) of node i, c_i is the community label of node i, and $\delta(c_i, c_j)$ is 1 if i and j are in the same community and 0 otherwise. The term $\frac{d_i d_j}{2m}$ approximates the expected number of edges (or expected weight) between i and j if edges are distributed randomly but respect node degrees. High modularity indicates that the actual number of intra-community edges is significantly above random expectation.

4.1.4 Modularity as a Stopping or Surgery Criterion

When using Ricci Flow for community detection, one can track how modularity evolves as edges get re-weighted and as one tries different thresholds for cutting. Typically, there is an intuitive sweet spot where further cutting does not substantially improve the modularity and might begin to over-segment the network.

A practical strategy is as follows:

1. Perform a fixed number of Ricci Flow iterations.

2. For a range of potential cut thresholds $T_1, T_2, \ldots, T_r$, remove edges with weight above T_j.
3. Compute modularity Q_j for each T_j.
4. Select the threshold T_j that yields the maximum Q_j preceding a drop in modularity.

This threshold selection process is akin to the notion of "neck pinches" or "surgeries" in the manifold setting: a large weight often signifies a bridging structure (negative curvature region grown large) that is "pinching off" from the main components. If an external ground-truth is known, one may prefer the threshold that simultaneously optimizes ARI and modularity. In the absence of external labels, maximizing modularity is a common choice to define the "best" partition.

4.1.5 Interpreting Network "Surgery" in This Context

A hallmark of the classical Ricci Flow with surgery on manifolds is that when the flow develops singularities (often visualized as "neck pinches"), the manifold is physically separated into topologically distinct pieces. Drawing an analogy, in discrete Ricci Flow on graphs, the edges that grow large (due to negative curvature) can be viewed as "singularities" or bridging regions, reminiscent of the neck that pinches in a continuous manifold. The act of removing these edges at some iteration is the direct analog of performing surgery on the manifold. After removing these "necks," the graph breaks into connected components, each presumably representing a dense or well-curved subregion, i.e., a community.

In practice, we carry out this network surgery step once or multiple times, balancing the preservation of meaningful connectivity with the desire to isolate truly separated clusters. Because many real networks can exhibit hierarchical or multi-level community structures, it is possible that each subcomponent can be further refined if we continue the process within it. This multiscale approach can be repeated if one suspects nested communities.

From a more theoretical vantage point, *why* does curvature—in the sense of Ollivier—detect communities so well? Intuitively, the geometry of a manifold with positive curvature is reminiscent of a cohesive, "ball-like" region, while negatively curved regions show hyperbolic expansions akin to branching. In network terms, cohesive subgraphs correspond to a "positive curvature signature" because local ran-

dom walks or local mass distributions align more easily, whereas bridging edges or tree-like expansions create a negative curvature effect.

4.2 Performance considerations

While the continuous Ricci Flow PDE can be computationally demanding in the smooth case, the discrete version has its own set of computational challenges. The major cost typically arises in Wasserstein Distances. Computing $W(m_x, m_y)$ for each edge (x, y). In a graph with n nodes and e edges, if we attempt an exact solution of the optimal transport problem, it can become computationally expensive. However, in many practical discrete settings, especially with local probability measures m_x that concentrate mass on immediate neighbours, $W(m_x, m_y)$ can often be computed with simpler combinatorial formulas or approximations [2].

Optimal Transportation Distance (OTD) and *Average Transportation Distance (ATD)* are the two main methods to quantify how easily or "cheaply" mass can be shifted from one node's local distribution to another's. In some formulations, one seeks the exact OTD, effectively solving the full Kantorovich transport problem between measures m_x and m_y. This yields the precise Wasserstein distance, but can be computationally intensive for large networks. An alternative is to approximate the cost by assigning a simpler, *average* cost of moving mass between neighbours, often referred to as the *average transportation distance* (ATD). In ATD, one typically does not compute a min-cost matching across all possible edges of transport but instead uses a simplified procedure (e.g., counting overlaps or partial overlaps of neighbourhoods) to estimate the transportation cost. Although ATD may lose some exactness compared to OTD, it greatly reduces the computational overhead and can still capture important curvature information in large-scale networks [15].

Despite these costs, modern computational resources and heuristics usually make discrete Ricci Flow feasible for networks of moderate size. For very large networks (millions of edges), one might rely on approximations, sampling techniques, or simplified versions of curvature (e.g., proxies to Ollivier Ricci curvature). In particular, using ATD rather than a full OTD calculation can strike a balance between accuracy and efficiency in such large-scale scenarios [15].

Chapter 5

The Developed Code

The objectives of the developed code were

- *Implementation of a Ricci Flow method* able to evaluate the Ollivier-Ricci curvature of a given graph and update its edges' weights accordingly. Then we wanted the method to perform surgery on weakly connected edges to allow for community detection.

  For these purposes we relied on Ollivier-Ricci library developed by Ni et al. [16], see section 5.1.

- *Plotting graphs and communities* to appreciate the behaviour of the method and obtain graphical results.

- *Testing the method* on synthetic graphs, trying to benchmark with an analogous test made by Ni et al. [2]. Results and further details are given in section 5.2.

- *Evaluation of performances* by comparing the method with other commonly used community detection methods.

- *Application on real world data*. To do this we chose Zachary's Karate Club graph, which is directly accessible from NetworkX library. Results and further details are given in section 5.3.

In this chapter we present the main ideas and tools related to the code. To get more insights on how the code as been constructed and subdivided (i.e., various classes and functions) we recommend consulting *CodeDocumentation* or the GitHub of the whole project: *RicciFlowNetwork*.

5.1 The setup

To facilitate the computation of Ricci curvature within networks, we made use of the GraphRicciCurvature library [16] as our starting point. This Python library is part of a more comprehensive library on Ricci curvature for networks. The latter provides tools to compute two discrete Ricci curvatures: Ollivier-Ricci and Forman-Ricci; it supports the analysis of both weighted and unweighted graphs. In addition, it offers basic methods for graph surgery and evaluation of the *adjusted rand index* (ARI).

Each graph we employed was generated with NetworkX library. We also used its methods as a starting point for graphical representations. For plotting we implemented `GraphDrawer` class which, among other functionalities, allows us to see the detected communities separated in subgraphs. Nodes are coloured with their corresponding community colour, giving a visual indication of method's performance.

We used this setup to implement the code workflow depicted in fig. 5.1.

5.2 Tests on Synthetic Graphs

5.2.1 Stochastic Block Model Test Graph

To start testing our code we choose a Stochastic Block Model (SBM) graph. An SBM graph is a type of random graph model used to generate networks with a predefined community structure. It is an extension of the Erdos–Rényi, where nodes are divided into communities, and the probability of an edge between two nodes depends on their community membership. It is characterized by N nodes, k communities and a connectivity matrix P_{ij} which represents the probability of an edge between a node in community i and a node in community j. Since communities are predefined when generating the graph, we can directly compare the detected communities with the true ones.

For our test we set

$$N = 500, \quad k = 2, \quad P = \begin{bmatrix} 0.20 & 0.03 \\ 0.03 & 0.20 \end{bmatrix}$$

with every initial weight equal to one.

Curvature values for the initial SBM graph are shown in fig. 5.2a, while on the right we have the updated curvatures after 10 iterations of Ricci Flow. We can see that the community structure of this graph is already well established; Ricci Flow has however

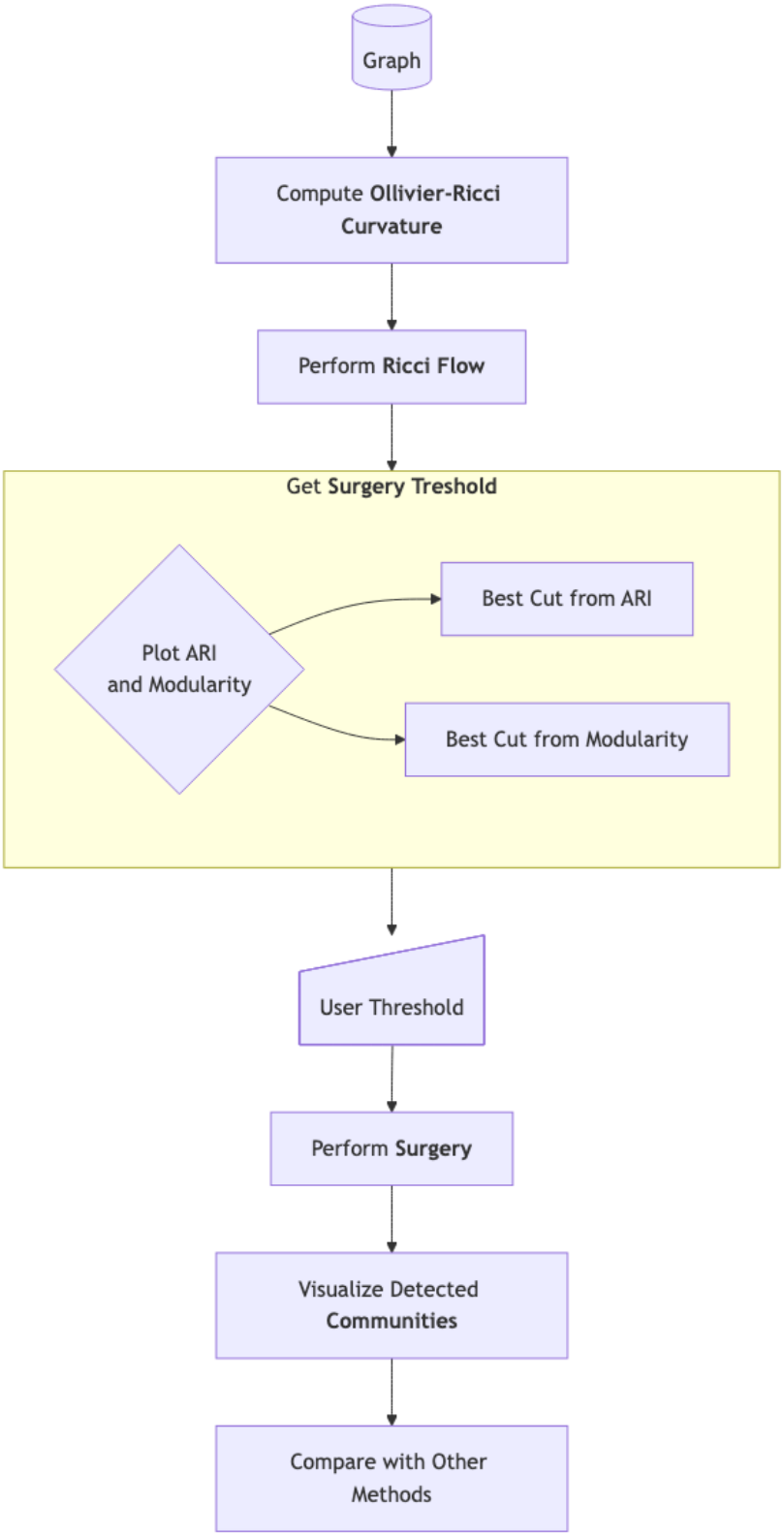
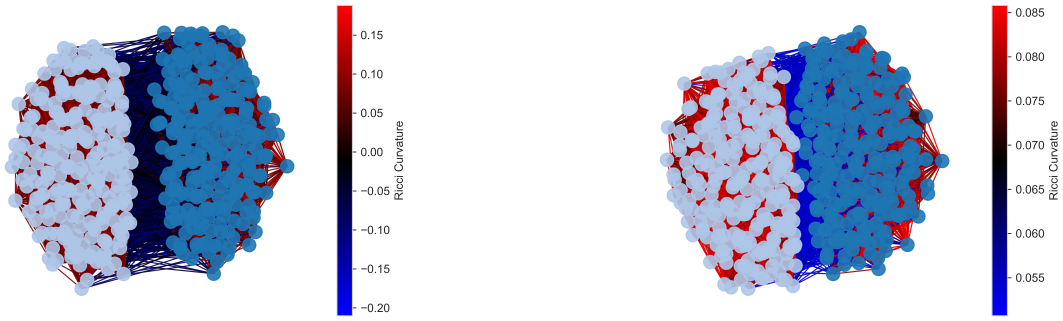


Figure 5.1 Code workflow.



(a) Initial SBM graph, before Ricci Flow.

(b) SBM graph after Ricci Flow.

Figure 5.2 Comparison of SBM graph before and after having applied 10 iterations of Ricci Flow on edges.

highlighted the weak connection of central inter community nodes as one can see in fig. 5.2b.

In fig. 5.3 we see that surgery with a cutoff between 1 and ≈1.5 leads to a perfect distinction between the two communities, i.e., an ARI of 1.

Lastly, fig. 5.4a depicts the graph after surgery with a cutoff between 1 and ≈1.5. As expected, we got a separation into two distinct clusters. Fig. 5.4b shows the two communities corresponding to the two connected components.

5.2.2 Lancichinetti-Fortunato-Radicchi Test Graph

As a more complex synthetic graph for testing we used a Lancichinetti-Fortunato-Radicchi (LFR) benchmark graph. LFR graphs are widely used for testing community-detection algorithms because they produce networks with heterogeneous (scale-free) degree distributions and community-size distributions, making them more realistic than simpler models. Nodes are assigned to communities according to specified power-law exponents, and a "mixing" parameter controls the fraction of edges that connect different communities.

For our test we built a graph with 500 nodes $(n = 500)$ with a degree distribution exponent $\tau_1 = 3$ and a community-size exponent $\tau_2 = 1.5$. For the mixing parameter we chose $\mu = 0.2$ indicates a relatively strong community structure by limiting the proportion of inter-community edges. We set each community to have a minimum of 20 nodes and a maximum of 70 nodes. The expected average degree is set to 20, with a maximum node degree capped at 50.

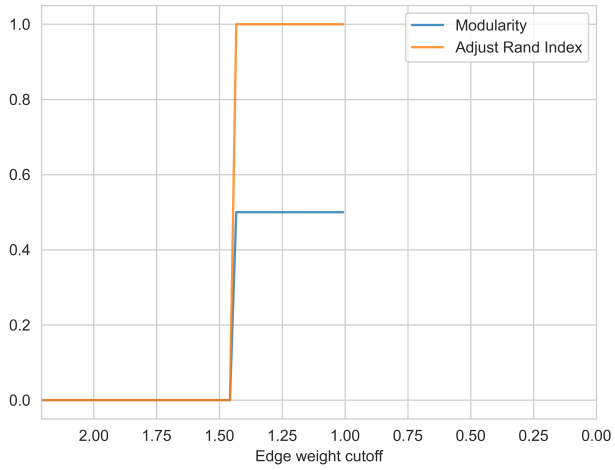
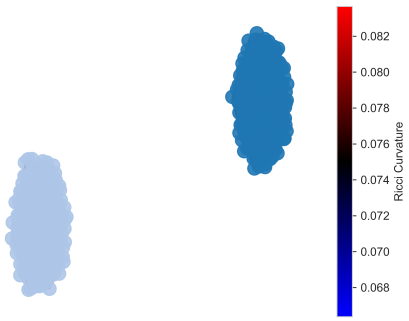
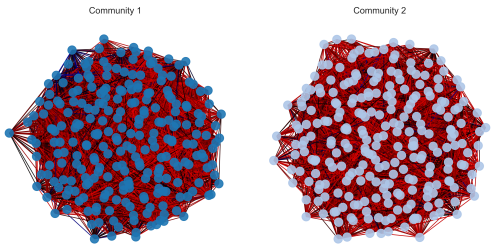


Figure 5.3 SBM graph's ARI and modularity behaviour for different surgery cutoffs.



(a) Final SBM graph, after surgery process.



(b) Detected communities after surgery on SBM graph.

Figure 5.4 Comparison of SBM graph after surgery and corresponding connected components (i.e., the detected communities).

(a) Initial LFR graph, before Ricci Flow.

(b) LFR graph after Ricci Flow.

Figure 5.5 Comparison of LFR graph before and after having applied 40 iterations of Ricci Flow on edges.

For this graph we applied 40 iterations of Ricci Flow as its structure is more complex than the previous SBM graph (see fig. 5.5a). In fig. 5.5b we see a convergence of the curvature values after Ricci Flow; the high number of nodes and communities makes difficult to visualize properly the graph, for this reason we introduced also a histogram plot of weights and curvature values in the final code implementation for the Karate club graph (see section 5.3).

Fig. 5.6 shows the behaviour of ARI and modularity depending on the chosen cutoff point. Also in this case we see that it is possible to obtain an ARI of 1.

In fig. 5.7 we have the graph after surgery (we chose as cutoff $\approx$1.12 to get an ARI of 1) and we can see that it got divided into connected components with all the nodes of the same colour, i.e., same community. In this case we did not produce a plot community by community due to the high number of clusters.

5.2.3 Additional Tests on Lancichinetti-Fortunato-Radicchi Graphs

As a final test for our code we decided to try reproducing a result presented in the work of Ni et al. (see *figure 8a* at page 12 of [2]). In particular, it is about applying Ricci Flow on different LFR graphs differing for the modularity parameter μ and the average degree. Of course one expects the ARI to decrease with increasing values of μ as the community structure becomes less defined.

In fig. 5.8 we present our results with OTD method for curvature evaluation. ARI decreases with increasing μ as expected and does not show significant differences between different values of average degree.

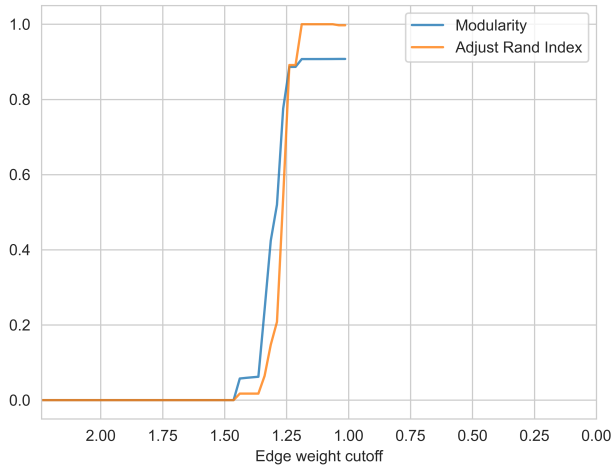


Figure 5.6 LFR graph's ARI and modularity behaviour for different surgery cutoffs.

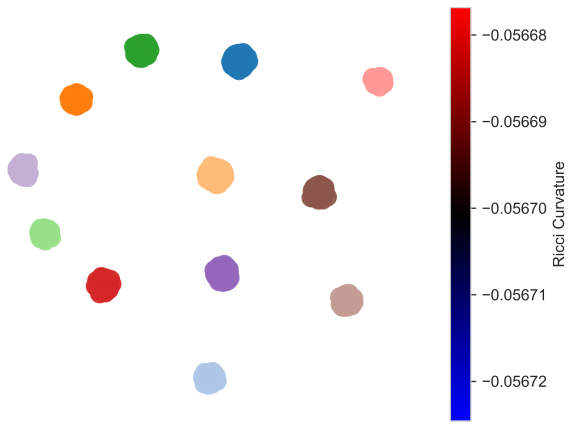


Figure 5.7 Final LFR graph, after surgery process.

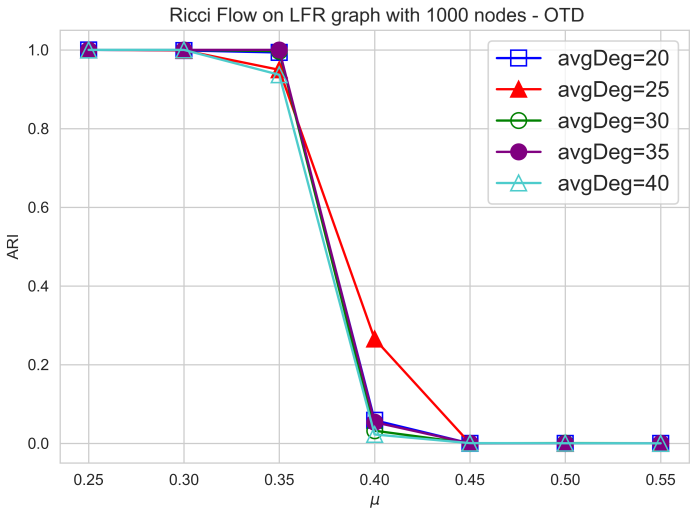


Figure 5.8 LFR OTD.

On the other hand, fig. 5.9 shows a much poorer performance and the choice of the average degree seems to be more relevant. This is in accordance with ATD method being faster than OTD, but also less precise.

Our graphs do not match exactly the plots of Ni et al., as for them the loss of ARI starts from higher values of μ. This could be for various reasons:

- It is unclear which parameter they used to generate LFR graphs. For example, using NetworkX it is not possible to control exactly the number of communities (only a minimum and maximum number can be set), so they probably generated their graphs differently, fixing some parameters we did not have complete control on.

- They used an average ARI, probably generating the same graph multiple times. We generated the graph and applied the Ricci Flow just once due to an already high execution time.

- Their surgery process is not specified in detail. They might have used multiple surgeries over the same Ricci Flow process to remove possible singularities affecting the final performance (even though this should not be the case for this kind of graph).

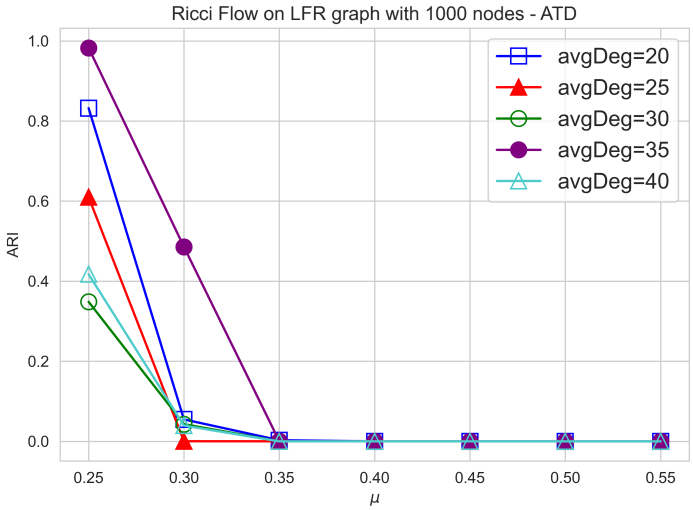


Figure 5.9 LFR ATD.

5.2.4 Execution Times

To conclude our testing part, we added some test performed on a LFR graph with fixed mixing parameter $\mu = 0.20$ and average degree of 20. We compared execution times for OTD and ATD methods applied to the same LFR graph, looking at the behaviour and stability for incresing iteration number in the Ricci Flow process. First we used a graph made up of n=500 nodes (fig. 5.10), in which we can see that ATD takes more iterations to converge and its performance is by no means comparable with the better OTD method. Even advantages in execution times for ATD are small and not consistent.

Then we set n=1000 and from fig. 5.11 we can see that the results are quite similar to the previous case but with an higher ARI for ATD method. Of course overall execution times increase with higher n as expected.

Lastly, we tested on a much bigger graph having n=2500, results are given in fig. 5.12. Here we see a significant difference in execution times when the Ricci Flow is performed with 30 iterations. ATD is more than a minute slower even thought its ARI has already converged. This is a bit unexpected but it might be due to the complexity of the graph with such a high number of nodes. Probably ATD does not succeed in making the graph structure converge and continues (unsuccessfully) updating curvatures consuming execution time. Again, the higher ARI of ATD for few iterations numbers can be explained as the method having updated the graph few times and so its original structure is still preserved at some level.

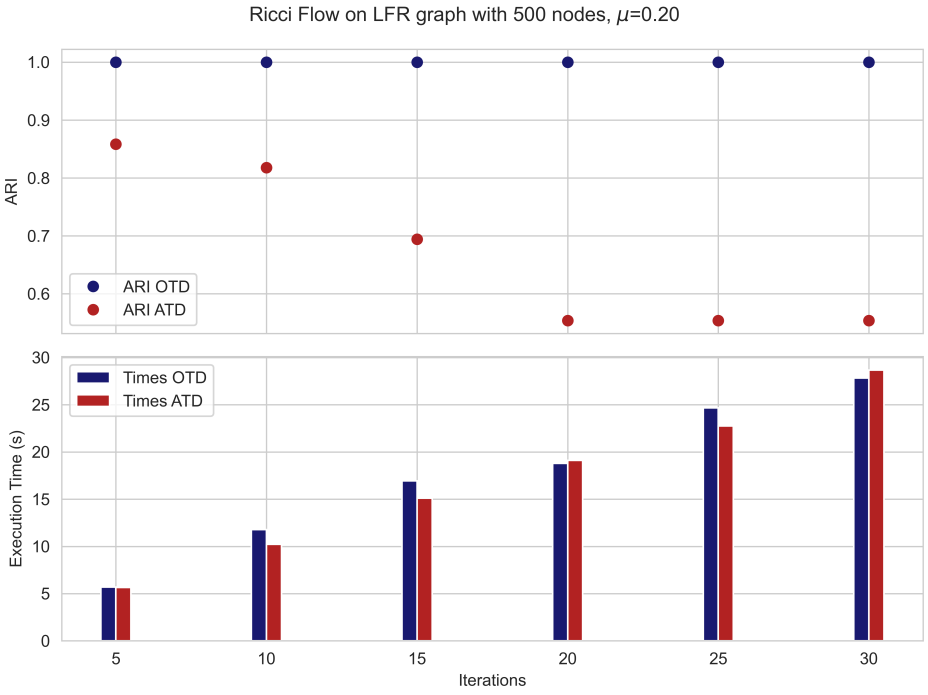


Figure 5.10 LFR execution times and ARI comparison between OTD and ATD methods for n=500.

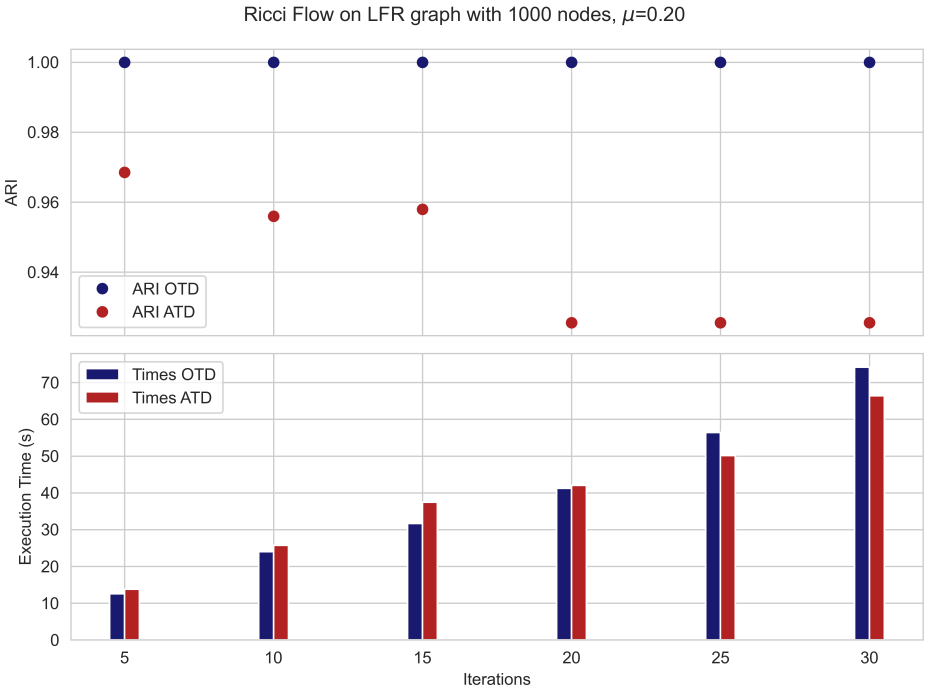


Figure 5.11 LFR execution times and ARI comparison between OTD and ATD methods for n=1000.

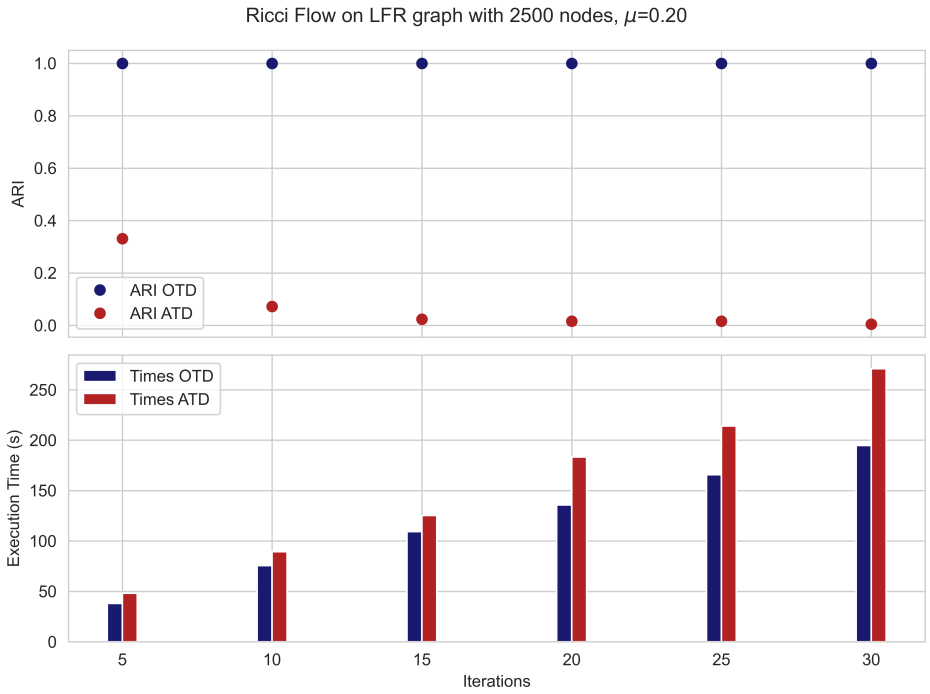


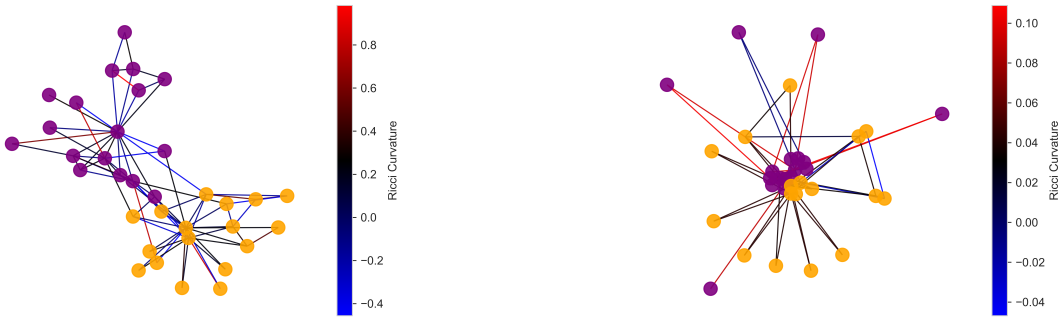
Figure 5.12 LFR execution times and ARI comparison between OTD and ATD methods for n=2500.

From this analysis we can say that ATD takes in general more iterations to converge and is more unstable if compared with OTD. As expected, ATD never performs better than OTD, what surprised us was ATD execution time, especially for the graph with $n = 2500$. We expected ATD to be faster with more consistence, even thought the behaviour could be explained as ATD is best suited for simple graph structures and thus might suffer when one has a graph with a high number of nodes.

5.3 Application to Zachary's Karate Club Graph

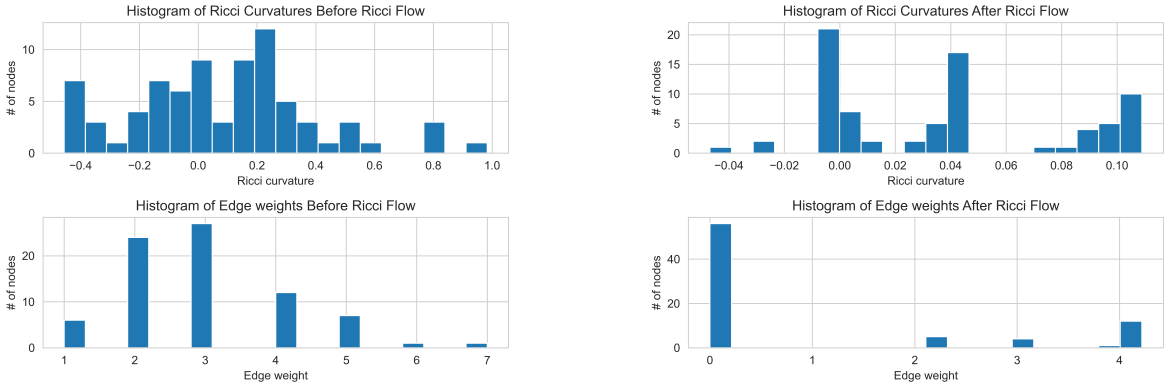
Zachary's Karate Club is a classic social network collected by Wayne Zachary in the 1970s [1]. It represents the relationships among 34 members of a university karate club, where each node is a person and an edge indicates some kind of social interaction (e.g., friendship or frequent contact) between two members. During the study, the karate club split into two factions due to an internal dispute, effectively dividing the network into two communities.

Therefore, our goal was to apply the Ricci Flow method to the graph in order to identify these two distinct communities. Fig. 5.13a is a plot of the initial graph as



(a) Initial Karate graph, before Ricci Flow. *(b)* Karate graph after Ricci Flow.

Figure 5.13 Comparison of Karate graph before and after having applied Ricci Flow on edges.



(a) Initial Karate graph, before Ricci Flow. *(b)* Karate graph after Ricci Flow.

Figure 5.14 Comparison of curvature values and weights before and after having applied Ricci Flow on edges.

loaded from NetwrokX, with edges' colours based on curvature values. In contrast, fig. 5.13b shows the same graph after Ricci Flow process, with updated curvature values.

As anticipated in the previous section, we added a histogram visualization for a better glimpse on curvatures and also on weights values. The original Ricci curvatures and weights can be seen in fig. 5.14a, while the updated ones are in fig. 5.14b.

In addition to plotting ARI and modularity with various cutoff thresholds, we also wanted to check the result produced by guessing the cut looking at modularity drops. In fig. 5.15 the "good cut" guessed as the one corresponding to the maximum before a modularity drop is depicted in red. As we hoped for, it also corresponds to

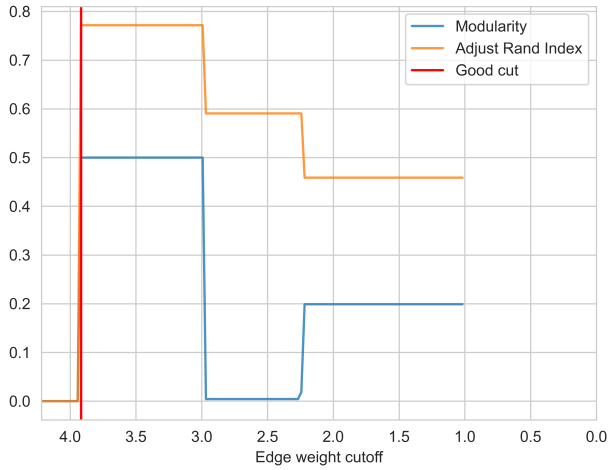


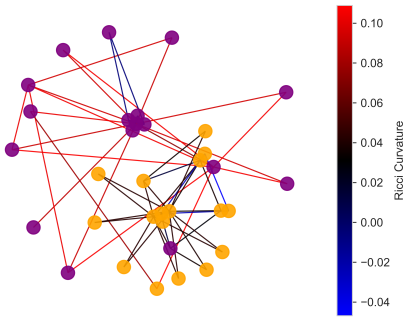
Figure 5.15 Karate Club graph's ARI and modularity behavior for different surgery cutoffs. In red is plotted the guessed "good cut" based on the maximum (preceding a drop) of modularity.

the maximum of ARI, confirming that the method can be still useful in cases in which ground truth is not available.

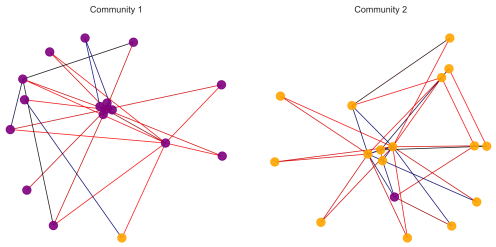
From fig. 5.16a we can see the graph after having perform surgery with the best cutoff giving an ARI of ≈0.772; which is almost the best result one can get for the Karate Club graph for most of the community detection algorithm [2]. To understand better the detected community structure we have them depicted in fig. 5.16b. Here we can see that only two nodes have been placed in the wrong community.

Lastly, we compared the results for ARI and modularity with other two popular community detection algorithms: *Louvian* and *Girvan-Newman*. Results are presented in fig. 5.17; for modularity obtained from Ricci Flow algorithm we mean the modularity value corresponding to the chosen cutoff for surgery (i.e., modularity at the end of Ricci Flow process, which we recall is evaluated using Louvian method).

Ricci Flow seems to have transformed the graph in such a way that its modularity has increased, it is higher than the one of the original graph detected by the two other methods. But perhaps the most significant result is that Ricci Flow methods yields a better ARI than Louvian method, which gives only ≈0.5, and the same value as Girvan-Newman.



(a) Final Karate Club graph, after surgery process.



(b) Detected communities after surgery on Karate Club graph.

Figure 5.16 Comparison of Karate graph after surgery and corresponding connected components (i.e., the detected communities).

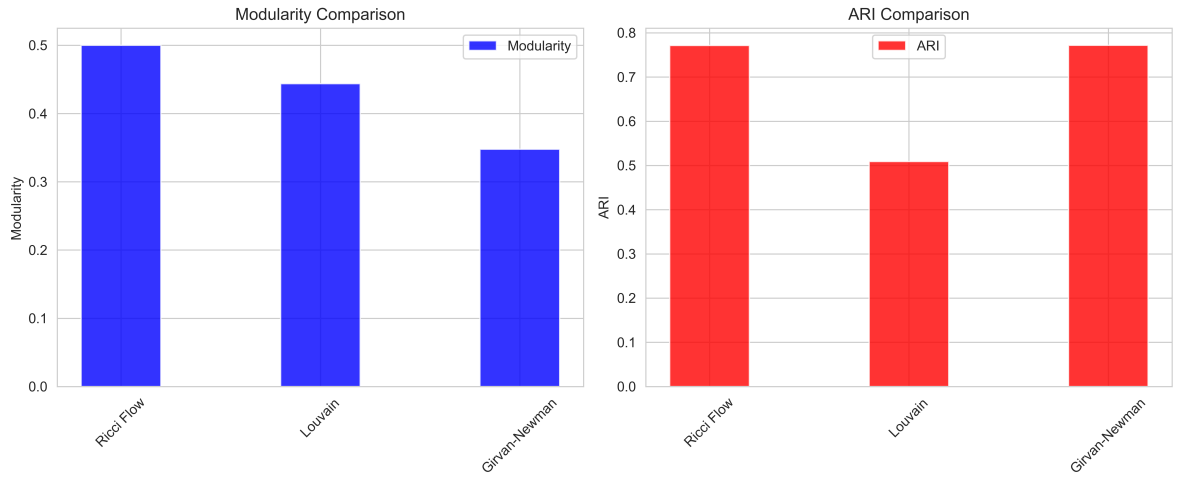


Figure 5.17 Karate Club graph comparison in performance between Ricci Flow, Louvian and Girvan-Newman methods.

REFERENCES

[1] Wayne W. Zachary. "An Information Flow Model for Conflict and Fission in Small Groups". In: *The University of Chicago Press Journals* (1997). URL: https://doi.org/10.1086/jar.33.4.3629752.

[2] Chien-Chun Ni et al. "Community Detection on Networks with Ricci Flow". In: (2019). arXiv: 1907.03993 [cs.SI]. URL: https://arxiv.org/abs/1907.03993.

[3] Lee J. M. *Introduction to Smooth Manifolds*. Second Edition. Springer, 2013. DOI: 10.1007/978-981-99-0565-2.

[4] Loring W. Tu. *An Introduction to Manifolds*. Second Edition. Springer, 2011. DOI: 10.1007/978-1-4419-7400-6.

[5] Lee J. M. *Introduction to Riemannian Manifolds*. Second Edition. Springer, 2018. DOI: 10.1007/978-3-319-91755-9.

[6] Richard H Bamler. "Recent developments in Ricci flows". In: (2021). arXiv: 2102.12615 [math.DG]. URL: https://arxiv.org/abs/2102.12615.

[7] B. Chow and D. Knopf. *The Ricci Flow: An Introduction*. American Mathematical Society, 2004. ISBN: 0-8218-3515-7.

[8] Grisha Perelman. "Ricci flow with surgery on three-manifolds". In: (2003). arXiv: math/0303109 [math.DG]. URL: https://arxiv.org/abs/math/0303109.

[9] Henri Poincaré. "Analysis Situs". Trans. by John Stillwell. In: *Journal de l'École Polytechnique* 2 (1895).

[10] Martin D. Crossley. *Essential Topology*. Springer, 2005. DOI: 10.1007/1-84628-194-6.

[11] Henri Poincaré. "Cinquième complément à l'analysis situs". Fifth Complement to Analysis Situs. Trans. by John Stillwell. In: *Rendiconti del Circolo Matematico di Palermo* 18 (1904 2008), pp. 45–110.

[12] Timo Weigand. "Introduction to String Theory". Lecture Notes. 2012.

[13] J. Polchinski. *String theory. Vol. 1: An introduction to the bosonic string*. Cambridge Monographs on Mathematical Physics. Cambridge University Press, Dec. 2007. DOI: 10.1017/CBO9780511816079.

[14] Mauro Carfora et al. "Ricci Flow from the Renormalization of Nonlinear Sigma Models in the Framework of Euclidean Algebraic Quantum Field Theory". In: (2019). arXiv: 1809.07652 [math-ph]. URL: https://arxiv.org/abs/1809.07652v2.

[15] Chien-Chun Ni et al. *Network Alignment by Discrete Ollivier-Ricci Flow*. 2018. arXiv: 1809.00320 [cs.SI]. URL: https://arxiv.org/abs/1809.00320.

[16] Ni C.C., Cruceru C., and Barrett S. *GraphRicciCurvature*. https://github.com/saibalmars/GraphRicciCurvature.git. 2019.