Hypergraph clustering using Ricci curvature: an edge transport perspective

Anonymous authors
Paper under double-blind review

Abstract

In this paper, we introduce a novel method for extending Ricci flow to hypergraphs by defining probability measures on the edges and transporting them on the line expansion. This approach yields a new weighting on the edges, which proves particularly effective for community detection. We extensively compare this method with a similar notion of Ricci flow defined on the clique expansion, demonstrating its enhanced sensitivity to the hypergraph structure, especially in the presence of large hyperedges. The two methods are complementary and together form a powerful and highly interpretable framework for community detection in hypergraphs.

1 Introduction

Hypergraphs are generalizations of graphs extending the concept of edges to encompass relationships involving any number of vertices rather than being restricted to pairs. Hypergraphs provide a more faithful data representation than graphs as many real-life interactions occur at a group level rather than at a binary one. Examples include social interactions, simultaneous participation at an event, co-authorship, mutual evolutionary traits, spatial proximity or molecular interactions. These interactions are better encoded by hypergraphs instead of traditional graphs, see Davis (1941); Dotson et al. (2022); Konstantinova and Skorobogatov (2001); Torres et al. (2021); Klamt et al. (2009). We refer to Berge (1984) and Bretto (2013) for a comprehensive overview of hypergraph theory.

Recently, machine learning tasks on hypergraphs have encountered a growing success, in particular for classification (Feng et al., 2019), regression (Liu et al., 2021), anomaly detection (Lee et al., 2022) or embedding (Zhang et al., 2019; Antelmi et al., 2023). The present article focuses on the clustering problem or community detection, i.e. recovering labels on the nodes in a fully unsupervised setting from a single hypergraph instance. Several approaches have been taken to tackle this problem such as embeddings using neural networks (Zhang et al., 2021; Lee and Shin, 2023), random walks-based embeddings (Hayashi et al., 2020; Huang et al., 2019) followed by a clustering in the Euclidean space. Some methods developed in Kumar et al. (2020); Kamiński et al. (2024) aim at maximizing hypergraph-modularity functions that measure the strength of a clustering. Finally, let us mention the work of Eyubov et al. (2023) that finds a partition in linear time by streaming all the nodes one by one.

The present work investigates generalizations of Ricci flow based clustering on hypergraphs. Ollivier-Ricci curvature, introduced by Ollivier (2007) for metric spaces and later adapted to graphs (Lin et al., 2011), defines a distance on graph edges that quantifies local curvature using optimal transport theory. Edges within strongly connected communities have a positive curvature while edges bridging two communities are negatively curved. Ricci curvature can be turned into a flow dynamic in order to further stress out this community structure and take some more global graph properties into account. Alternative notions of curvature for graph processing are also present in Forman (2003); Sreejith et al. (2016); Iyer et al. (2024); Tian et al. (2025). Ricci curvature on graphs has been used to derive theoretical bounds on the spectrum of the graph Laplacian by Lin and Yau (2010) and Bauer et al. (2011). For applied purposes, it has been used for data exploration (Ni et al., 2015), representation learning (Zhang et al., 2023), topological data analysis

(Carriere and Blumberg, 2020; Hacquard and Lebovici, 2024), and in particular clustering (Ni et al., 2019), relying on the property of the Ricci flow to place emphasis on the community structure.

There has been a few attempts to generalize Ollivier-Ricci curvature to hypergraphs, in particular by Coupette et al. (2022). The authors introduce a notion of curvature using random walks on the nodes. As pointed out by Chitra and Raphael (2019), in most practical cases, considering a random walk on the hypergraph is equivalent to a random walk on its *clique expansion* (a graph representation where each hyperedge is replaced by a weighted clique). Doing so is a common way to circumvent the complex hypergraph structure by replacing it with a graph with a similar structure. However, as noted in Chitra and Raphael (2019), this reduction can result in significant information loss, as distinct hypergraphs may share identical clique graphs (Hein et al., 2013). In addition, replacing a single hyperedge between k nodes implies adding up to $\binom{k}{2}$ connections between nodes which can be computationally prohibitive in a network with mutual interactions between many agents. Extensions of Ricci curvature on directed hypergraphs have also been proposed by Eidi and Jost (2020).

The main contribution of this work aims at defining a notion of Ricci curvature on hypergraphs where we consider probability distributions on the edges instead than on the nodes. This approach leverages the *line expansion* of the hypergraph, which is a graph representation where nodes correspond to hyperedges, and edges reflect hyperedges intersections. The line expansion is a common way to represent a hypergraph for learning purposes, see Bandyopadhyay et al. (2020); Yang et al. (2022). In addition, it has been demonstrated in Kirkland (2018) that almost all the hypergraph information is retained by the joint knowledge of both its clique and line expansions. The current work puts a particular focus on clustering. We perform a thorough experimental study on synthetic and real data comparing the approach where we transport measures on the edges to the more standard approach where we transport measures on the nodes. More precisely, this notion of edge transport should be favored when we have a large number of small communities, when the edges between communities have a larger cardinality than the ones within communities, and more generally for hypergraphs with very large hyperedges as this approach is much more efficient computationally.

The work is organized as follows: Section 2 introduces the foundational concepts of hypergraph theory and Ollivier-Ricci curvature on graphs. Section 3 presents two different possible expansions of Ricci curvature for hypergraphs: the standard one where we transport measures on the nodes, and the main contribution of this paper where we transport measures on the edges. We also compare the two approaches theoretically on a synthetic example. Section 4 provides a detailed baseline of experiments on synthetic data and on real data where we compare both methods to state-of-the-art clustering algorithms on hypergraphs, along with a computational analysis of both methods.

2 Model

This section presents the main concepts related to hypergraph analysis and the hypergraph partitioning problem. We also investigate standard methods to replace hypergraphs by traditional graphs and the loss of information it implies.

2.1 Hypergraphs and associated graphs

Hypergraphs, definitions *Undirected hypergraphs*, sometimes referred to as *hypernetworks* or *2-modes networks*, consist of a set of nodes V and a set of *hyperedges* E. Extending the concept of graphs where edges link two distinct nodes, hyperedges are defined as non-empty sets of nodes of arbitrary size. For a hyperedge $e = (u_1, \dots, u_k)$, its cardinality k is referred to as its *size*. A hypergraph in which all hyperedges have the same size k is called a $k-$*uniform hypergraph*. Notably, graphs are $2-$uniform hypergraphs. We further define the star of a vertex $v \in V$ as the set of hyperedges that include v: $St(v) = \{e \in E | v \in e\}$.

Clique expansion Given a hypergraph H with node set $V = (v_1, \dots, v_n)$ and edge set $E = (e_1, \dots, e_m)$, its *incidence matrix* $\mathbf{I} \in \{0, 1\}^{n \times m}$ is defined as $\mathbf{I}_{i,j} = 1$ if $v_i \in e_j$ and 0 otherwise. This representation as a rectangular matrix makes further analysis of hypergraphs much more complicated than that of graphs, which can be described by their adjacency matrix. A commonly used simplification of hypergraphs is the

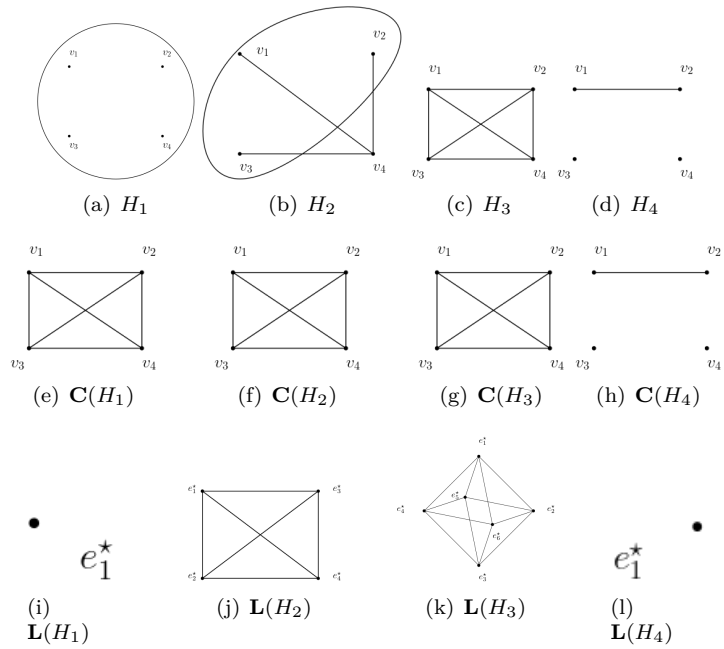


Figure 1: Examples of hypergraphs H_i with their clique $\mathbf{C}(H_i)$ and line $\mathbf{L}(H_i)$ expansions. H_1, H_2 and H_3 share the same clique but have different line expansions.

clique expansion, where each edge is replaced by a clique. More precisely, the clique expansion $\mathbf{C}(H)$ of the hypergraph H is the graph with node set V and with an edge (x, y) if there exists a hyperedge $e \in E$ such that $x, y \in e$. The clique expansion provides a convenient simplification of hypergraphs by grouping nodes belonging to the same edge. However, there is a certain loss of information since multiple hypergraphs share the same clique expansion, see Figure 1. We refer to Hein et al. (2013) for a more thorough analysis of differences between hypergraphs and their clique expansion. Notably, according to Chitra and Raphael (2019) and Hayashi et al. (2020), unless the nodes are assined an *edge-dependent weighting*, standard notions of random walks and Laplacians on the hypergraphs can be expressed on a reweighted clique expansion. This implies that directly applying Laplacian-based methods, such as clustering, on hypergraphs often incurs the information loss inherent in their clique expansions. We note that the adjacency matrix A_C of the clique graph can be derived from the incidence matrix via $A_C = \mathbf{I}\mathbf{I}^T - D_V$ where D_v is the diagonal node-degree matrix, with entries $d_i = \sum_{j=1}^{n} \mathbf{I}_{i,j}$.

Line expansion Another graph associated with H is the *line graph* $\mathbf{L}(H)$. Each edge in H now corresponds to a node in $\mathbf{L}(H)$. Thus, $\mathbf{L}(H)$ has node set $(e_1^\star, \ldots, e_m^\star)$ and we have an edge $(e_i^\star, e_j^\star)$ in $\mathbf{L}(H)$ if the corresponding hyperedges intersect in H, i.e. $e_i \cap e_j \neq \emptyset$. Similarly to the clique expansion, many hypergraphs share the same line expansion, see Figure 1. However, joint knowledge of both the line and clique expansions seems to capture most of the structural information of the hypergraph. Indeed in Figure 1, hypergraphs H_1, H_2 and H_3 all share the same clique expansion while having different line expansions (they even have different node sets). Similarly to the clique graph, the adjacency matrix A_L of the line graph can be derived from the incidence matrix via $A_L = \mathbf{I}^T\mathbf{I} - D_e$ where D_e is the diagonal matrix of edge-lengths, with entries $\delta_i = |e_i|$. Let us define the dual hypergraph $H^\star = (E^\star, V^\star)$ as the hypergraph obtained by swapping the edge and node sets, i.e. $E^\star = (e_1^\star, \ldots, e_m^\star)$ and $V^\star = (v_1^\star, \ldots v_n^\star)$ where $v_i^\star = \{e_j^\star | v_i \in e_j\}$. We can easily observe, see Zhou et al. (2022), that the clique expansion is the line graph of the dual hypergraph: $\mathbf{C}(H) = \mathbf{L}(H^\star)$.

Gram mates We can wonder if the joint knowledge of both the clique graph and the line graph allows to reconstruct the hypergraph. The answer is negative, as there exists pairs of distinct 0-1 matrices (A, B) such that $AA^T = BB^T$ and $A^TA = B^TB$. Such matrices are called *Gram mates*, and we refer to Kim and Kirkland (2022) for a proof of this result and a thorough analysis of families of matrices that generate Gram mates. However, it has been demonstrated in Corollary 1.1.1 of Kirkland (2018) that if we uniformly pick

two $m \times n$ 0-1 matrices, the probability that they are Gram mates decays exponentially as $(m, n) \to \infty$. Thus, for practical applications, the information loss from replacing a hypergraph with its clique and line expansions together is negligible. This representation also simplifies hypergraph analysis significantly.

Clustering We consider hypergraphs where nodes are associated with discrete labels $\{1, \ldots, K\}$, representing communities or clusters. Intuitively, such hypergraphs exhibit a community structure, with denser connections between nodes sharing the same label compared to those with different labels. We assume a fully unsupervised setting, where we try to infer the labels (up to permutation) simply via the hypergraph structure. This generalization of the graph partitioning problem has been studied extensively, and we refer to Çatalyürek et al. (2023) for a survey of different approaches to tackle this problem. A popular method to determine the quality of a partitioning is given by *modularity* maximization, initially defined by Newman (2006). We recall its definition in the graph case. For a graph $G = (V, E)$ and a partitioning $\mathbf{C} = (C_1, \ldots, C_K)$ of V into K subsets, we define the *modularity function* Q as:

$$Q(\mathbf{C}) = \sum_{i=1}^{K} \frac{e_G(C_i)}{|E|} - \sum_{i=1}^{K} \left(\frac{vol(C_i)}{vol(V)} \right)^2,$$

where $e_G(C_i)$ is the number of edges in the subgraph of G induced by a node set C_i, and $vol(A) = \sum_{v \in A} deg(v)$ is the *volume* of any subset A of nodes. Up to renormalization, the modularity function computes the difference between the number of edges uncut by the partitioning and the expected number of edges the same partitioning would yield in the Chung-Lu random graph model introduced in Chung and Lu (2006). Maximizing Q over every partitioning yields partitions that minimize cuts within communities, aligning with the underlying structure. There have been several works generalizing modularity for hypergraphs, notably Kumar et al. (2020) and Kamiński et al. (2024), by generalizing the Chung-Lu random model to hypergraphs, and allowing different weights to hyperedges of different sizes. Modularity also serves as a ground truth-free measure of the quality of a given clustering.

2.2 Ricci curvature and flows on graphs

This paper addresses the clustering problem using an extension of Ollivier-Ricci curvature to hypergraphs. This method extends prior works on Ollivier-Ricci curvature, initially defined in Ollivier (2007) on metric spaces, and extended to the case of graphs in Lin et al. (2011). In Ni et al. (2019), the Ricci curvature is turned into a discrete flow such that edges between communities are heavily curved. They directly derive a clustering algorithm. We start by recalling the key concepts of their method in the case of graphs.

Discrete Ricci curvature Let $G = (V, E)$ be a weighted graph with a weight function $w : E \to \mathbb{R}_{\geq 0}$. In what follows, the function w is set to be a *dissimilarity measure* between nodes. Let d denote the shortest path distance between nodes induced by weights w. We adopt the framework introduced in Ni et al. (2019): for each node x, we define a probability measure $\nu_x^{\alpha,p}$ over x and its neighbors given by:

$$\nu_x^{\alpha,p}(y) = \begin{cases} \alpha & \text{if } y = x, \\ \frac{1-\alpha}{C} \exp(-d(x,y)^p) & \text{if } y \in \mathcal{N}(x), \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where $\mathcal{N}(x)$ denotes the set of neighbors of x. Here, $C = \sum_{y \in \mathcal{N}(x)} \exp(-d(x,y)^p)$ is a normalization constant to ensure that $\nu_x^{\alpha,p}$ is a probability measure. The parameters $\alpha \in [0, 1]$ and p respectively denote the amount of mass retained at x and how much we want to penalize far neighbors. When not required or clear from context, we will drop the superscript and simply write $\nu_x^{\alpha,p} = \nu_x$.

The *discrete Ricci curvature* κ is a function defined on the edges of the graph such that for an edge $e = (x, y)$:

$$\kappa(x, y) = 1 - \frac{W(\nu_x, \nu_y)}{d(x, y)},$$

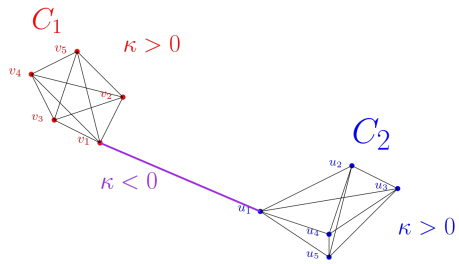


Figure 2: Ollivier-Ricci curvature on a graph with a clear two-communities structure.

where W is the Wasserstein distance between probability measures for the cost function d, see Santambrogio (2015). More precisely, a discrete transport plan T between the measures ν_x and ν_y is an application $T : V \times V \to [0, 1]$ such that $\sum_{v' \in V} A(u, v') = \nu_x(u)$ and $\sum_{u' \in V} A(u', v) = \nu_y(v)$. The Wasserstein distance corresponds to the total cost of moving ν_x to ν_y with the optimal transport plan for a cost function d, i.e.

$$W(\nu_x, \nu_y) = \inf \left\{ \sum_{u,v \in V} T(u, v) d(u, v) | T \text{ transport plan between } \nu_x \text{ and } \nu_y \right\}.$$

The sign of κ provides information on the density of connections. Indeed, two nodes x and y belonging to the same community will typically share many neighbors, hence transporting ν_x to ν_y has a smaller cost than moving x to y and therefore induces a positive curvature. Conversely, nodes belonging to different communities do not share many neighbors, such that moving ν_x to ν_y requires traveling through the edge (x, y) bridging the two communities. This in turn corresponds to a negative curvature. We refer to Figure 2 for a concrete example: when transporting ν_{v_i} to ν_{v_j} and ν_{u_i} to ν_{u_j}, most of the mass remains on the nodes such that measures are transported at a low cost. In the extreme case where $\alpha = 1/5$, all the mass remains and $\kappa(v_i, v_j) = \kappa(u_i, u_j) = 1$. When transporting ν_{v_1} to ν_{u_1}, all the v_i's must be transported to u_i's through the edge (u_1, v_1) such that the total transport cost is larger than $d(u_1, v_1)$ and therefore $\kappa(u_1, v_1) < 0$.

It instantly appears that trimming edges of low curvature naturally partitions the graph where each connected component corresponds to a different community. We note that Ricci curvature can theoretically be computed for any pair of nodes in the graph. However, for our purposes, we only need to compute it for adjacent nodes.

Discrete Ricci flow and community detection To amplify the impact of heavily negatively curved edges and incorporate more global graph structure, we turn Ricci curvature into a discrete flow that modifies the graph weights. We recall that the edge weights are dissimilarities, and a high weight corresponds to a high distance between nodes. The initial weight between x and y is $w^{(0)}(x, y) = w(x, y)$. We iteratively update the weights with:

$$w^{(l+1)}(x, y) = \left(1 - \kappa^{(l)}(x, y)\right) w^{(l)}(x, y), \tag{2}$$

where $\kappa^{(l)}$ is the Ricci curvature computed with the distance induced by the weights $w^{(l)}$. The updated weights after the l-th iteration are referred to as the *Ricci flow*. This flow dynamic stretches edges of low curvature, making it more and more costly to transport measures between communities. The complete clustering algorithm is described by Algorithm 1, and simply trims edges with a high Ricci flow. The choice of the number of flow iterations N is mostly heuristic, we refer to Ni et al. (2019) and the corresponding `python` library `GraphRicciCurvature` for some guidelines. The threshold τ at which to trim the weights is a crucial parameter and can also be taken as the one that maximizes the *modularity function* defined in Section 2.1, which evaluates clustering quality without ground truth labels.

This approach ensures that inter-community connections are down-weighted while preserving intra-community coherence. This enables a robust and interpretable method for clustering. Indeed, the method relies on direct computations of the Ollivier-Ricci curvature which is strongly associated to an intrinsic community partitioning of the graph, see Figure 2. We defer an extended comparison of the interpretability of Ricci-based clustering as opposed to neural network embeddings to Section 4.2.

Input: Graph $G = (V, E)$, Number of iterations N, Threshold τ
Output: Clustering labels Y
for $i \leftarrow 1$ ***to*** N **do**
  Update the weights using Equation 2;
end
Trim all edges with a weight larger than τ to create a new graph $\tilde{G}$;
Compute connected components $C_1 \coprod \ldots \coprod C_k$ of $\tilde{G}$;
for $i \leftarrow 1$ ***to*** $|V|$ **do**
  Set $Y_i = j$ if $v_i \in C_j$;
end

Algorithm 1: Ricci flow algorithm.

3 Two complementary notions of Ricci curvature on hypergraphs

Extending Ricci curvature to hypergraphs presents two main challenges:

- Ricci curvature is traditionally defined for pairs of nodes and needs to be extended to hyperedges.
- Computing transport distances between measures defined on the nodes of a hypergraph typically relies on the clique expansion (see Section 2.1 and Hayashi et al. (2020)). As a consequence, this approach introduces a loss of structural information, as clique expansions fail to capture certain higher-order interactions.

The first challenge is straightforward to treat, for instance by taking the average or the maximum of all pairwise curvatures of nodes in a hyperedge (as in Coupette et al. (2022)). We will discuss different possibilities in Section 3.1. The second challenge constitutes the main contribution of this paper and will further be developed in Section 3.2 where we explore a method to transport hyperedges.

Note that we restrict our analysis to the Ollivier-Ricci curvature. Another notion, the *Forman-Ricci curvature* (Forman, 2003), has been extended to the case of hypergraphs in Leal et al. (2021). We refer to Samal et al. (2018) for a detailed comparison of these two curvature types in the graph case.

3.1 Nodes transport on the clique graph

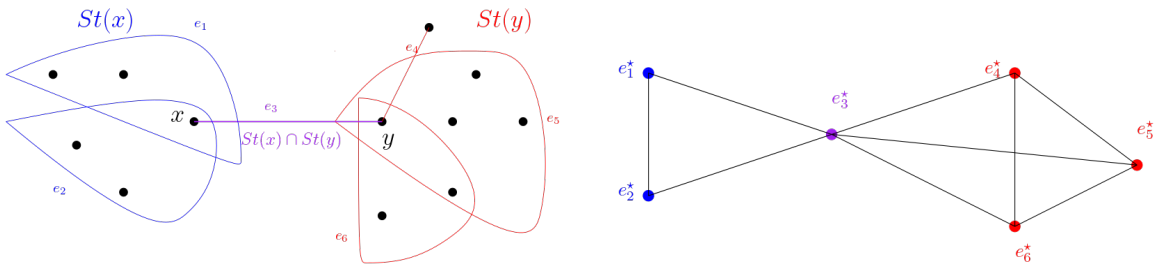
Clique graph transport Let H be a hypergraph with nodes $V = (v_1, \ldots, v_n)$ and edges $E = (e_1, \ldots, e_m)$. To compute the Ricci curvature between two nodes x and y, we can still define measures ν_x and ν_y on x and y and their neighbors following Equation 1, as the definition of a neighbor is unchanged. The first solution is simply to replace the hypergraph with its clique graph. Two possibilities are considered to obtain a clique graph from a hypergraph. For the sake of clarity, we assume the hypergraph to be unweighted. Let $A_C = (a_{ij})_{1 \leq i,j \leq n}$ be the adjacency matrix of the clique expansion of H.

(i) The first option is to consider a simple unweighted clique graph expansion where

$$a_{ij} = \begin{cases} 1 & \text{if there exists } e \in H \text{ such that } (v_i, v_j) \subset e, \\ 0 & \text{otherwise.} \end{cases}$$

(ii) The second option is proposed by Zhou et al. (2022) and relies on a Jaccard index weighting. Recall that for two finite sets U and V, the Jaccard index $J(U, V) = \frac{|U \cap V|}{|U \cup V|}$ is a measure of similarity between sets U and V. Let $H^\star = (V^\star, E^\star)$ be the dual hypergraph as defined in 2.1. We remark that for two nodes v_i and v_j, $a_{ij} = 0$ if and only if $|v_i^\star \cap v_j^\star| = 0$. The weight of two adjacent nodes in the clique graph is given by $1/J(v_i^\star, v_j^\star)$. The Jaccard index is inverted to provide a dissimilarity weighting.

A discussion on the relative performance of these two weightings can be found in Section 4.3.



(a) Stars of x and y

(b) Corresponding line graph

Figure 3: Edge-Ricci transport between nodes x and y. A measure on $St(x)$ is transported onto a measure on $St(y)$ via the line graph.

Aggregating pairwise curvatures After computing the Ricci flow $w_{\mathcal{N}}^{(l)}(e)$ for every pair of adjacent nodes in the clique graph of H, the Ricci weight of a given hyperedge $e \in E$ is obtained by aggregating all pairwise flows between nodes of e:

$$w_{\mathcal{N}}^{(l)}(e) = \underset{v_i, v_j \in e}{Agg} \left(w^{(l)}(v_i, v_j) \right),$$

where Agg is an aggregation function. Experimentally, we found that aggregating using the *maximum* of all pairwise curvatures systematically yielded the best clustering performance.

The clustering pipeline given by Algorithm 1 then operates similarly by trimming all hyperedges with a Ricci flow larger than some threshold. The optimal threshold can be found by either maximizing the modularity of the clique graph, or a corresponding notion of *hypergraph modularity*, see Kamiński et al. (2024).

3.2 Edges transport on the line graph

As discussed in 2.1, replacing a hypergraph by its clique graph induces a substantial loss of information. To address this, we propose a novel notion of Ricci curvature specific to hypergraphs. Instead of transporting measures defined on the nodes of a hypergraph, we transport measures defined on its edges. The motivation is that two nodes from different communities should be contained in sets of hyperedges that typically do not intersect much.

Edge-Ricci curvature The proposed method also computes pairwise curvature and then extends it to hyperedges using an aggregation step similar to the one previously defined. We first describe the Ricci curvature computation for a hypergraph $H = (V, E)$ with dissimilarity weights $d : E \to \mathbb{R}_{>0}$. Let x and y be two adjacent nodes in the hypergraph. We define simple probability measures μ_x and μ_y on their stars $St(x)$ and $St(y)$:

$$\mu_x(e) = \begin{cases} \frac{d(e)}{C} & \text{if } e \in St(x) \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

where $C = \sum_{e \in St(x)} d(e)$ is a normalization constant to ensure that μ_x is a probability measure. The definition of μ_y is analogous. Note that unlike the measures $\nu_x^{\alpha,p}$ defined on the nodes, the measures μ_x do not depend on any hyper-parameter, simplifying their use.

The distance d_H between two edges e and f is defined using the line graph: $d_H(e, f) = d_{LG}(e^\star, f^\star)$ where d_{LG} is the standard shortest path distance for graphs, see Figure 3. Note that similarly to the clique graph, there are several ways the line graph can be weighted (for instance using Jaccard weights between hyperedges). This aspect will be discussed in more details in Section 4.3. Similarly to the graph case, we can then define an optimal transport plan and a corresponding Wasserstein distance W between the two measures μ_x and μ_y. The *edge-Ricci curvature* $\kappa_{\mathcal{E}}$ between two nodes x and y is therefore defined as

$$\kappa_{\mathcal{E}}(x,y) = 1 - \frac{W(\mu_x, \mu_y)}{\max\limits_{e \in St(x) \cap St(y)} d(e)}.$$

Pairs of nodes in different communities tend to have very different stars, which implies in turn a low curvature. Conversely, if the two nodes x and y belong to the same community, the measures μ_x and μ_y can be transported onto each other with a low cost, resulting in a curvature close to 1. We remark that even in the case of a 2-uniform hypergraph, this definition is not equivalent to the usual curvature on graphs from Section 2.2. In addition, we lose the interpretability observed in Figure 2 concerning the sign of the curvature.

Edge-Ricci flow Similarly to the graph case, the curvature is then turned into a discrete flow. We start with an initial edge weighting $w_{\mathcal{E}}^{(0)} = d$. At the l-th iteration, the weights are updated according to the following three steps:

(i) Compute the Ricci curvature $\kappa_{\mathcal{E}}^{(l)}$ for every pair of adjacent nodes in the weighted hypergraph.

(ii) For every hyperedge e, aggregate all pairwise curvatures with an aggregation function Agg to define the curvature of the hyperedge, similarly to Section 3.1:

$$\kappa_{\mathcal{E}}^{(l)}(e) = \underset{v_i, v_j \in e}{Agg} \left(\kappa_{\mathcal{E}}^{(l)}(v_i, v_j) \right).$$

(iii) For every hyperedge e, update the edge weights using the Ricci flow dynamic:

$$w_{\mathcal{E}}^{(l+1)}(e) = \left(1 - \kappa_{\mathcal{E}}^{(l)}(e)\right) w_{\mathcal{E}}^{(l)}(e).$$

Similarly to the graph case this procedure is iterated N times, and hyperedges with a weight larger than a given threshold are trimmed. The resulting connected components form the assigned node communities. In traditional Ricci flow with $p = 0$, negatively curved edges gradually gain weight, increasing the cost of moving probability measures through them. Here, we take a dual perspective: edges with low curvature are assigned increasing mass, making them progressively more expensive to transport. The line graph, however, remains unchanged, such that distances between edges are unaffected by the flow process. We refer to Tian et al. (2025) for another example of method using the line graph for community detection.

3.3 A toy clustering example

We start by comparing the node-Ricci flow from Section 3.1 with the edge-Ricci flow from Section 3.2 in a synthetic example where transport maps and curvatures can be computed explicitly. We adopt a very similar setting to the one of Ni et al. (2019) where we consider a family of hypergraphs $H(a, b)$ where $a, b \geq 3$ are integers. Consider b complete binary graphs $C_1, \ldots, C_b$ on a distinct vertices. For each complete graph C_i, take a particular vertex v_i (called gateway vertex) and consider the hyperedge of size b, $(v_1, \ldots v_b)$ connecting all gateway vertices across the communities. This hypergraph exhibits a clear community structure where each C_i forms a distinct community, and there is a single hyperedge connecting all the communities together. Figure 4 provides a visual representation.

The case of the node-Ricci flow for an unweighted clique expansion has been treated explicitly in Ni et al. (2019), for measures defined by Equation 1 with $\alpha = 0$ and $p = 0$. They observe that by symmetry, in the clique expansion, the Ricci flow at the $l-$th iteration can only take three possible values: $w_{\mathcal{N},1}^{(l)}$ for edges between two gateway nodes, $w_{\mathcal{N},2}^{(l)}$ for edges between a gateway node and another node from its community, and $w_{\mathcal{N},3}^{(l)}$ for edges between two non-gateway nodes from the same community. They further demonstrate that if $a > b$, $w_{\mathcal{N},3}^{(l)} = \left(\frac{1}{a}\right)^l \underset{l \to \infty}{\to} 0$ and that there exists $\lambda > \frac{1}{a}$ and constants $c_1 > c_2$ such that $w_{\mathcal{N},1}^{(l)} = c_1 \lambda^l + o(\lambda^l)$ and $w_{\mathcal{N},2}^{(l)} = c_2 \lambda^l + o(\lambda^l)$ as $l \to \infty$. This implies that the Ricci flow-based trimming algorithm properly identifies the communities $C_1, \ldots, C_b$.

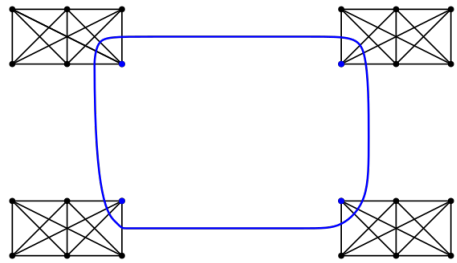


Figure 4: Example of hypergraph $H(a, b)$ with $a = 6$ and $b = 4$. The gateway nodes are represented in blue.

We derive a similar result for specific measures μ. Equation 3 is slightly modified such that the transported measures are now defined, for two nodes x and y as:

$$\tilde{\mu}_x(e) = \begin{cases} 0 & \text{if } x, y \in e \\ \frac{d(e)}{C} & \text{if } e \in St(x) \text{ and } e \notin St(y) \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

where C is a normalization constant. In practice, this choice of measure yields results extremely similar to the ones defined by Equation 3. It simply discards edges shared between x and y, which would typically not play a big role in deriving the transport plan. This choice simply ensures a much simpler expression for the Ricci flow to facilitate the delivery of the following proposition:

Proposition 1 *We keep the same notation as above and consider the edge-Ricci flow with measures $\tilde{\mu}$ defined by Equation 4 and aggregation Agg with the maximum function. At the l-th iteration, the Ricci flow can only take three possible values: $w^{(l)}_{\mathcal{N},1}$ for the edge of size b connecting all gateway nodes, $w^{(l)}_{\mathcal{N},2}$ for binary edges between a gateway node and another node from its community, and $w^{(l)}_{\mathcal{N},3}$ for binary edges between two non-gateway nodes from the same community. Under these circumstances, we have that for all $l > 0$:*

$$\begin{cases} w^{(l)}_{\mathcal{N},1} = 2, \\ w^{(l)}_{\mathcal{N},2} = \frac{4+(a-2)w^{(l-1)}_{\mathcal{N},2}}{2+(a-2)w^{(l-1)}_{\mathcal{N},2}} \in [1, 2], \\ w^{(l)}_{\mathcal{N},3} = 1. \end{cases}$$

The proof is deferred to Appendix A. Proposition 1 implies that the edge-Ricci flow can also identify the community structure with an edge cut-off parameter between 1 and 2. We notice that in this case, the explicit expression of the Ricci flow, although much simpler than in the node-Ricci case, does not have an exponential decay to 0 for the intra-community edges. This implies that the cut-off parameter might be more delicate to tune in practice. However, note that Proposition 1 holds for any values of a and b, contrary to the result of Ni et al. (2019) which only holds for $a > b$. This implies that in practice, edge-Ricci allows for better detection of many small communities.

4 Experiments

To evaluate the practical effectiveness of Ricci flow for hypergraph clustering, we conducted a range of experiments on synthetic and real data. We opted for a Jaccard type weighting for the clique graph and the line graph. Curvatures of hyperedges are computed by aggregating curvatures of pairs of nodes using the maximum function. For node-Ricci flow, we have used the python library `GraphRicciCurvature` with parameters $p = 1$ and $\alpha = 0.5$ for the measures $(\nu_x^{\alpha,p})_{x \in V}$ defined on the nodes. We refer to Section 4.3 for a discussion about the choice of default values for the hyperparameters. The clustering accuracy is measured using the *Normalized Mutual Information score* (NMI). The implementations of edge-Ricci and node-Ricci flows for hypergraphs are available at `https://github.com/AnonRicciHG/Ricci_curv_HG` and are directly adapted from the `GraphRicciCurvature` library.

4.1 Hypergraphs stochastic block models

We first evaluate our method on a synthetic set-up to highlight differences between edge-Ricci and node-Ricci flows. We consider a hypergraph stochastic block model inspired by Kim et al. (2018). We consider a hypergraph with n nodes and k communities. We give ourselves two size parameters s_{in} and s_{out} and generate N_{in} hyperedges of size s_{in} between nodes of the same community for each community and N_{out} hyperedges of size s_{out} containing at least one node of each community. In the graph case where $s_{in} = s_{out} = 2$, this is a form of stochastic block model, see Abbe (2018). This is a simple model where intra-community edges can be interpreted as signal and inter-community edges as noise. In the hypergraph case, as soon as $s_{in} \geq 3$, inter-community edges are also regrouping nodes of the same community together and cannot simply be interpreted as pure noise.

We try to reconstruct the communities using edge-Ricci and node-Ricci flows clustering algorithms. We consider randomly generated hypergraphs of $n = 100$ nodes, $k = 2$ equal-sized communities and compare the performances of the two methods for various values of s_{in} and s_{out}. We fix the value of N_{in} and measure the evolution of the clustering accuracy for a growing number N_{out} of "noisy" hyperedges. We compute $N = 20$ flow iterations for each method and take the thresholding parameter $\tau^\star$ that maximizes the NMI. We average over 5 random hypergraph generations and report the corresponding NMI in Figure 5. We remark that node-Ricci flow seems to perform comparatively better than edges-flow when the size of intra-community edges s_{in} is large and the size of inter-community edges s_{out} is small and conversely. These differences can be simply understood heuristically: when contaminating a hypergraph by adding a hyperedge of size s, the clique graph will have up to $s(s-1)/2$ new edges while the line graph will only have a single new additional node. This accounts for an improved robustness of edges-flow to large hyperedges contamination, as it mostly relies on shortest-path computations on the line-graph.

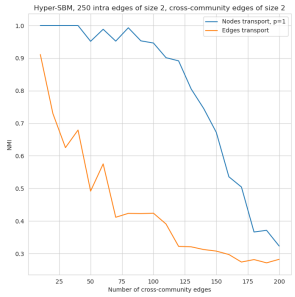
4.2 Real data

We further compare the clustering performance of nodes and edges-flow with state-of-the-art graph and hypergraph clustering methods.

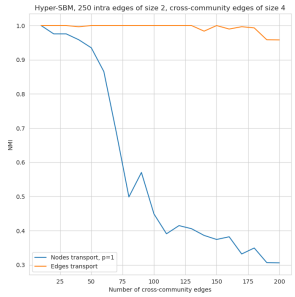
Methodology and data We evaluate our method on the datasets presented in Table S.1. A detailed description of the datasets can be found in Section B. Data and code to reproduce the experiments are available at `https://github.com/AnonRicciHG/Ricci_curv_HG`. We adopt the same benchmark study as in Lee and Shin (2023). For each dataset, we iterate Ricci flows $N = 20$ times. The edge-cutting threshold τ is taken as the one that maximizes the hypergraph modularity, computed with the `hypernetx` python package.

Quantitative results We report the NMI of our method in Table 1, where we compare ourselves with graph and hypergraph partitioning state-of-the-art methods. Node2vec (Grover and Leskovec, 2016), DGI (Veličković et al., 2018) and GRACE (Zhu et al., 2020) embed the hypergraph's clique expansion in a Euclidean space using various neural networks architectures. S^2-HHGR from Zhang et al. (2021) and TriCL from Lee and Shin (2023) directly embed the hypergraphs using a neural network architecture. Communities are then detected using a k-means algorithm. Note that for TriCL and S^2-HHGR, additional node-feature information is used to improve the prediction, while Ricci flow methods rely only on the hypergraph structural information. All scores for these methods are reported from Lee and Shin (2023). We also report the score obtained by the modularity maximization algorithm from Kamiński et al. (2021). OOM indicates an out-of-memory error on a 24GB GPU (reported from Lee and Shin (2023)). OOT indicates that the results could not be obtained within 72 hours on a single 8-core laptop with a Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz processor unit.

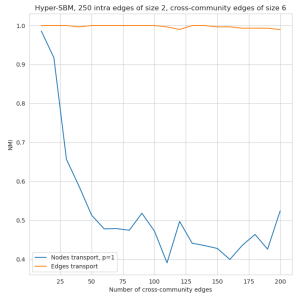
Ricci-based clustering methods provide an overall competitive accuracy, with a better performance than simple graph vectorization methods, while managing to be as competitive as state-of-the-art neural network-based hypergraph embeddings on a few datasets. In particular, we achieve a perfect clustering on the `Zoo` dataset using edge-Ricci flow. This dataset stands out for having very few nodes and edges and demonstrates the ability of Ricci-based methods to capture fine structural information on small graphs. We assert that a



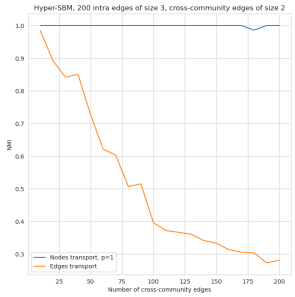
(a) $s_{in} = 2, s_{out} = 2$



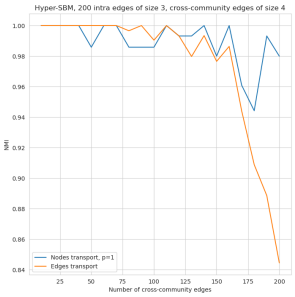
(b) $s_{in} = 2, s_{out} = 4$



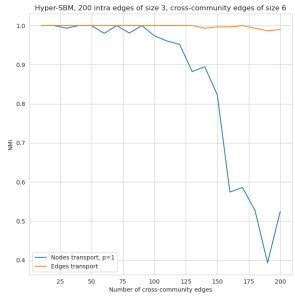
(c) $s_{in} = 2, s_{out} = 6$



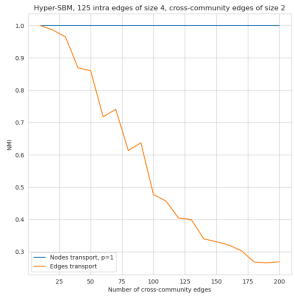
(d) $s_{in} = 3, s_{out} = 2$



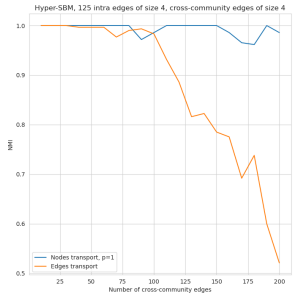
(e) $s_{in} = 3, s_{out} = 4$



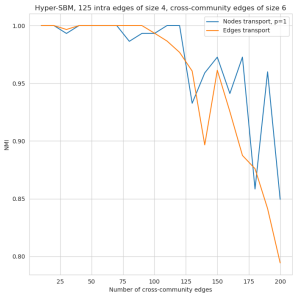
(f) $s_{in} = 3, s_{out} = 6$



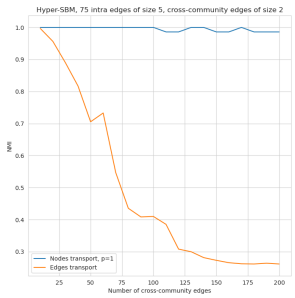
(g) $s_{in} = 4, s_{out} = 2$



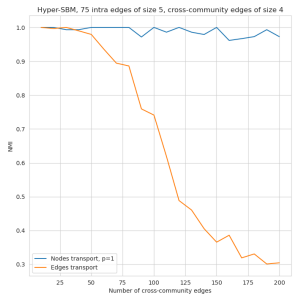
(h) $s_{in} = 4, s_{out} = 4$



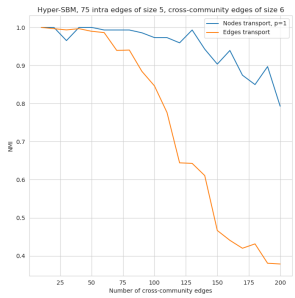
(i) $s_{in} = 4, s_{out} = 6$



(j) $s_{in} = 5, s_{out} = 2$



(k) $s_{in} = 5, s_{out} = 4$



(l) $s_{in} = 5, s_{out} = 6$

Figure 5: Hypergraph stochastic block model reconstruction using hypergraph notions of Ricci flow.

Dataset	Cora-C	Cora-A	Citeseer	Pubmed	Zoo	Mushroom	NTU2012
Node2Vec	39.1	16.0	24.5	23.1	11.5	1.6	78.3
DGI	**54.8**	45.2	40.1	30.4	13.0	OOM	79.6
GRACE	44.4	37.9	33.3	16.7	7.3	OOM	74.6
S^2-HHGR	51.0	45.4	41.1	27.7	90.9	18.6	82.7
TriCL	54.5	**49.8**	**44.1**	**30.0**	91.2	3.8	**83.2**
Modularity	45.0	33.4	33.8	25.0	77.7	**43.4**	74.5
N-Ricci	45.8	39.4	38.8	27.8	96.2	OOT	76.3
E-Ricci	43.3	39.1	38.3	22.7	**100.0**	42.3	77.6

Table 1: NMI clustering accuracy on real datasets.

significant advantage of our method is its high level of explainability, particularly when compared to methods based on neural network embeddings. Specifically, Ricci flow facilitates clustering on graphs with several notable properties:

- It is grounded in established theoretical works on metric spaces, see Ollivier (2007), and benefits from convergence results when applied on random geometric graphs, as shown in Van Der Hoorn et al. (2021).
- The method is computable in closed-form on simple examples as demonstrated in Ni et al. (2019) and further elaborated in Section 3.3.
- It is strongly associated with the intuitive notion of a community on a graph, by assigning a substantial weight to edges across communities, see Figure 2.
- The method relies on explicit and transparent computations and is fully deterministic.

However, it is to be noted that when extended to hypergraphs, some of these points fail to be as compelling, notably the theoretical guarantees and the interpretability of the sign of the curvature from Figure 2. Nevertheless, our method's explicit computation of a flow process, which draws a strong analogy to its well-established geometric counterpart, offers greater interpretability compared to neural network approaches. The latter often involves many parameters, making their interpretation challenging.

4.3 Hyperparameter discussion

Both notions of Ricci curvature on hypergraphs depend on a moderate number of hyperparameters. Some of them can significantly influence the final performance. In this section, we review all the hyperparameters, conduct a sensitivity analysis, and provide guidelines for tuning.

The number of flow iterations N This parameter is inherited from Ricci flow on graphs. The effect of this parameter has been examined in Section 4.1 of the Supplementary Material of Ni et al. (2019). The authors illustrate the convergence of the edge weights as N tend to infinity. Therefore, choosing a large N therefore has no detrimental impact aside from increasing the running time. In the associated code, the authors observe convergence for N between 20 and 50. This guideline still holds in our case, and we did also observe a stabilization of the Ricci flow weights for N larger than 20.

Parameters α and p for $\nu^{\alpha,p}$ These parameters are used to define the measures in Equation 1. These parameters are directly inherited from Ricci flow on graphs. The impact of the parameter p has been studied in Section 4.2 of the Supplementary Material of Ni et al. (2019). We nevertheless decide to review the impact of this parameter in Figure S.2 where we observe that in some cases, there is a drop of performance for $p = 0$ while $p = 1$ and $p = 2$ show a similar performance. Additionally, we decide to investigate the role of the parameter α in some critical cases, see S.3. This study has not been conducted in loc. cit. where the authors advocate for a default choice of $\alpha = 0.5$. Although it is challenging to provide specific guidelines, extreme values of the parameter α can lead to serious instabilities. As a result, taking α close to 0.5 is a safe choice.

Aggregation and graph weightings Both methods compute a Ricci-curvature between pairs of adjacent nodes which is in turn aggregated at the hyperedge level, as suggested in Coupette et al. (2022). We have explored two choices: taking the maximum and the arithmetic average.

In addition, the transport costs depend on the initial clique graph and line graph weightings discussed in Section 3.1. We compare between unweighted graphs and Jaccard weightings. We have studied the joint impact of these two parameters and report the results in Figures S.4 and S.5. We make the following observation:

- For node-Ricci flow, the clique graph weighting has little to no impact, while using the maximum of the pairwise curvature for the hyperedge curvature consistently yields better results.
- For edge-Ricci flow, we observe a very important difference between these parameters. In short, the combination Jaccard-maximum is the most competitive when $s_{in} \leq s_{out}$ and Uniform-average is the most competitive when $s_{in} > s_{out}$. The other two combinations always yield an intermediate accuracy. This implies that an intersection-sensitive line graph weighting has a strong positive impact in the case of large hyperedges between communities. Nevertheless, we observe in Figure S.6 that edge-Ricci flow with any combination of parameters is always outperformed by a node-Ricci flow with a maximum aggregation function in the case where $s_{in} > s_{out}$, as already observed in Section 4.1. Therefore, we always recommend using the Jaccard-maximum parameter combination for the edge-Ricci flow to fully leverage its potential in the case of large inter-community hyperedges.

Joint study of every hyperparameter on a real dataset In the previous paragraphs, we considered critical cases where a change in parameters could have a dramatic impact. We now study the robustness of the two types of transport applied to a real dataset. In Tables S.2 and S.3, we examine the joint effect of every parameter except α and τ on the dataset `NTU2012`. We observe that for this real dataset, the choice of weighting has almost no impact on the final accuracy. However, using the maximum function for Agg provides a much better accuracy than the average. Finally, We observe a modest improvement by considering $p = 1$ over $p = 0$.

Threshold parameter τ Both notions of Ricci flow simply operates a community-sensitive reweighting of the edges, equivalent to a hierarchical clustering of the nodes. The cut-off parameter τ enables access to a particular clustering, making it a crucial parameter when comparing to ground-truth labels. To chose the value of τ, we compare the one that maximizes the hypergraph modularity τ_H, the one that maximizes the graph modularity on the clique expansion τ_C and the one giving the clustering with the highest NMI, i.e. closest to the ground-truth one $\tau^\star$. We report these results in Tables S.4 and S.5. We observe that using τ_H gives results relatively close from the optimal ones using $\tau^\star$ and better accuracy than using τ_C. In particular, on the `Zoo` dataset, $\tau_H = \tau^\star$ for both notions of transport, while τ_C gives extremely poor results. This additional experiment aligns with the findings of Kamiński et al. (2021) and Lee and Shin (2023), advocating for using hypergraph-specific tools when available, rather than reducing to the clique expansion.

Finally, note that there is not necessarily a single best value for the parameter τ. Indeed, modifying the optimal τ can provide clusters coarser or finer than the ground-truth, which can be appealing in some applications.

4.4 Computational comparison

We observe in Table 1 that node-Ricci flow failed to provide results for the `Mushroom` dataset in a reasonable time. Even after 72 hours, not a single Ricci-flow iteration was completed, despite this dataset containing a relatively small number of hyperedges (see Table S.1). To better understand this limitation, we further investigate the computational complexity of both methods.

Computing the Ricci curvature of all edges in a hypergraph $H = (V, E)$ requires solving $|\mathcal{E}|$ optimal transport problems where $\mathcal{E}$ is the edge set of the clique expansion $\mathbf{C}(H)$. In practice, the optimal transport costs between two measures with at most n points are computed using the Sinkhorn algorithm from Sinkhorn (1974), which provides an ε-approximation at a cost $O(n^2/\varepsilon^2)$ according to Lin et al. (2019). This implies

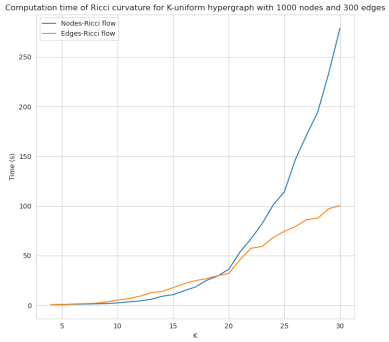


Figure 6: Computation times of Ricci curvature for both methods on a K-uniform hypergraph as a function of K.

worst-case complexities of order $O\left(|\mathcal{E}| \times \max_{x \in V} |\mathcal{N}(x)|^2/\varepsilon^2\right)$ for node-Ricci and $O\left(|\mathcal{E}| \times \max_{x \in V} |St(x)|^2/\varepsilon^2\right)$ for edge-Ricci. In turn, large hyperedges have a stronger impact on the node-Ricci complexity.

To justify these theoretical claims, we consider a K-uniform hypergraph with 1000 nodes and 300 edges of length K drawn uniformly at random. We report the computational times in seconds for a single Ricci-flow interation in Figure 6. Computations are carried out for unweighted clique and line expansions and are averaged over five different hypergraph initializations. We observe that the computational cost of nodes-Ricci curvature indeed blows up for hypergraphs with large hyperedges due to the inflated neighborhood size $|\mathcal{N}(x)|$. Edge-Ricci flow demonstrates much slower growth in computational time, remaining efficient even for hypergraphs with large hyperedges. Computations are executed on a single 8-core laptop with a Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz processor unit. These computations explain why only edge-Ricci could produce results in a reasonable time on the `Mushroom` dataset. Although this dataset has relatively few hyperedges, the hyperedges are exceptionally large, inducing severe computational challenges for node-Ricci flow. The computational times from Figure 6 along with the quantitative results from Table 1 indicate that Ricci-flow based methods are in particular more suited to dealing with rather small hypergraphs.

4.5 Wrap-up, comparison of Node-Ricci and Edge-Ricci flows

Based on the findings from Sections 3.3 and 4, node-Ricci and edge-Ricci flows are complementary approaches for competitive hypergraph clustering. Indeed, as the first one primarily leverages the clique expansion and the second one the line expansion information, taken together, they capture most of the information of the hypergraph. More precisely, edge-Ricci is preferable to node-Ricci whenever:

- There are many small communities.
- The intra-community edges are typically smaller than inter-community edges.
- The hypergraph has very large hyperedges, inducing prohibitive computations on the clique graph.

When dealing with real data with no structural a priori on the communities, both methods tend to perform comparatively, see Section 4.2. However, they can have sensibly different computational costs making one preferable over the other depending on the structure of the hypergraph, in particular the number of hyperedges and their size. We note that alternating between iterations of node-Ricci flow and edge-Ricci flow did not provide any substantial quantitative benefit.

Conclusion

We have developed two methods to extend Ricci flow to hypergraphs. The first method, node-Ricci flow, applies standard Ricci flow on the clique expansion and further aggregates it to hyperedges. The second

method, edge-Ricci flow, is the main contribution of this article and constitutes an original approach to transporting edges using the line expansion. Both methods define new weights on the hyperedges, which can then be simply turned into a partitioning algorithm. Each method has its own advantages depending on the specific characteristics of the hypergraph.

A natural extension of this work would be to consider a co-optimal transport of both nodes and edges, as discussed in Titouan et al. (2020); Chowdhury et al. (2024). Additionally, the notion of discrete Ricci flow has more practical uses beyond graph partitioning. For instance, it can be used as a method to preprocess hypergraphs by generating new weights that better highlight underlying community structures, as seen in Khodaei et al. (2024). Finally, several notions of curvature have been developed for graph, as explored in Iyer et al. (2024); Cushing et al. (2022). Further extending these notions to hypergraph data constitutes a potential research direction.

References

Abbe, E. (2018). Community detection and stochastic block models: recent developments. *Journal of Machine Learning Research*, 18(177):1–86.

Antelmi, A., Cordasco, G., Polato, M., Scarano, V., Spagnuolo, C., and Yang, D. (2023). A survey on hypergraph representation learning. *ACM Computing Surveys*, 56(1):1–38.

Bandyopadhyay, S., Das, K., and Murty, M. N. (2020). Hypergraph attention isomorphism network by learning line graph expansion. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 669–678. IEEE.

Bauer, F., Jost, J., and Liu, S. (2011). Ollivier-ricci curvature and the spectrum of the normalized graph laplace operator. *arXiv preprint arXiv:1105.3803*.

Berge, C. (1984). *Hypergraphs: combinatorics of finite sets*, volume 45. Elsevier.

Bretto, A. (2013). Hypergraph theory. *An introduction. Mathematical Engineering. Cham: Springer*, 1.

Carriere, M. and Blumberg, A. (2020). Multiparameter persistence image for topological machine learning. *Advances in Neural Information Processing Systems*, 33:22432–22444.

Çatalyürek, Ü., Devine, K., Faraj, M., Gottesbüren, L., Heuer, T., Meyerhenke, H., Sanders, P., Schlag, S., Schulz, C., Seemaier, D., et al. (2023). More recent advances in (hyper) graph partitioning. *ACM Computing Surveys*, 55(12):1–38.

Chitra, U. and Raphael, B. (2019). Random walks on hypergraphs with edge-dependent vertex weights. In *International conference on machine learning*, pages 1172–1181. PMLR.

Chowdhury, S., Needham, T., Semrad, E., Wang, B., and Zhou, Y. (2024). Hypergraph co-optimal transport: Metric and categorical properties. *Journal of Applied and Computational Topology*, 8(5):1171–1230.

Chung, F. R. and Lu, L. (2006). *Complex graphs and networks*. Number 107. American Mathematical Soc.

Coupette, C., Dalleiger, S., and Rieck, B. (2022). Ollivier-ricci curvature for hypergraphs: A unified framework. *arXiv preprint arXiv:2210.12048*.

Cushing, D., Kamtue, S., Liu, S., and Peyerimhoff, N. (2022). Bakry-émery curvature on graphs as an eigenvalue problem. *Calculus of Variations and Partial Differential Equations*, 61(2):62.

Davis, G. (1941). Gardner,(1941). *Deep South*.

Dotson, G. A., Chen, C., Lindsly, S., Cicalo, A., Dilworth, S., Ryan, C., Jeyarajan, S., Meixner, W., Stansbury, C., Pickard, J., et al. (2022). Deciphering multi-way interactions in the human genome. *Nature Communications*, 13(1):5498.

Eidi, M. and Jost, J. (2020). Ollivier ricci curvature of directed hypergraphs. *Scientific Reports*, 10(1):12466.

Eyubov, K., Faraj, M. F., and Schulz, C. (2023). Freight: Fast streaming hypergraph partitioning. *arXiv preprint arXiv:2302.06259*.

Feng, Y., You, H., Zhang, Z., Ji, R., and Gao, Y. (2019). Hypergraph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 3558–3565.

Forman (2003). Bochner's method for cell complexes and combinatorial ricci curvature. *Discrete & Computational Geometry*, 29:323–374.

Grover, A. and Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864.

Hacquard, O. and Lebovici, V. (2024). Euler characteristic tools for topological data analysis. *Journal of Machine Learning Research*, 25(240):1–39.

Hayashi, K., Aksoy, S. G., Park, C. H., and Park, H. (2020). Hypergraph random walks, laplacians, and clustering. In *Proceedings of the 29th acm international conference on information & knowledge management*, pages 495–504.

Hein, M., Setzer, S., Jost, L., and Rangapuram, S. S. (2013). The total variation on hypergraphs-learning on hypergraphs revisited. *Advances in Neural Information Processing Systems*, 26.

Huang, J., Chen, C., Ye, F., Wu, J., Zheng, Z., and Ling, G. (2019). Hyper2vec: Biased random walk for hyper-network embedding. In *Database Systems for Advanced Applications: DASFAA 2019 International Workshops: BDMS, BDQM, and GDMA, Chiang Mai, Thailand, April 22–25, 2019, Proceedings 24*, pages 273–277. Springer.

Iyer, R., Wang, Y., Wang, W., and Sun, Y. (2024). Non-euclidean mixture model for social network embedding. *Advances in Neural Information Processing Systems*, 37:111464–111488.

Kamiński, B., Misiorek, P., Prałat, P., and Théberge, F. (2024). Modularity based community detection in hypergraphs. *Journal of Complex Networks*, 12(5):cnae041.

Kamiński, B., Prałat, P., and Théberge, F. (2021). Community detection algorithm using hypergraph modularity. In *Complex Networks & Their Applications IX: Volume 1, Proceedings of the Ninth International Conference on Complex Networks and Their Applications COMPLEX NETWORKS 2020*, pages 152–163. Springer.

Khodaei, M., Bidabad, B., Shiri, M. E., Sedaghat, M. K., and Amirifard, H. (2024). Classification of alzheimer's disease using ricci flow-based spherical parameterization and machine learning techniques. *Signal, Image and Video Processing*, 18(10):6529–6545.

Kim, C., Bandeira, A. S., and Goemans, M. X. (2018). Stochastic block model for hypergraphs: Statistical limits and a semidefinite programming approach. *arXiv preprint arXiv:1807.02884*.

Kim, S. and Kirkland, S. (2022). Gram mates, sign changes in singular values, and isomorphism. *Linear Algebra and its Applications*, 644:108–148.

Kirkland, S. (2018). Two-mode networks exhibiting data loss. *Journal of Complex Networks*, 6(2):297–316.

Klamt, S., Haus, U.-U., and Theis, F. (2009). Hypergraphs and cellular networks. *PLoS computational biology*, 5(5):e1000385.

Konstantinova, E. V. and Skorobogatov, V. A. (2001). Application of hypergraph theory in chemistry. *Discrete Mathematics*, 235(1-3):365–383.

Kumar, T., Vaidyanathan, S., Ananthapadmanabhan, H., Parthasarathy, S., and Ravindran, B. (2020). A new measure of modularity in hypergraphs: Theoretical insights and implications for effective clustering. In *Complex Networks and Their Applications VIII: Volume 1 Proceedings of the Eighth International Conference on Complex Networks and Their Applications COMPLEX NETWORKS 2019 8*, pages 286–297. Springer.

Leal, W., Restrepo, G., Stadler, P. F., and Jost, J. (2021). Forman–ricci curvature for hypergraphs. *Advances in Complex Systems*, 24(01):2150003.

Lee, D. and Shin, K. (2023). I'm me, we're us, and i'm us: Tri-directional contrastive learning on hypergraphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 8456–8464.

Lee, G., Choe, M., and Shin, K. (2022). Hashnwalk: Hash and random walk based anomaly detection in hyperedge streams. *arXiv preprint arXiv:2204.13822*.

Lin, T., Ho, N., and Jordan, M. (2019). On efficient optimal transport: An analysis of greedy and accelerated mirror descent algorithms. In *International Conference on Machine Learning*, pages 3982–3991. PMLR.

Lin, Y., Lu, L., and Yau, S.-T. (2011). Ricci curvature of graphs. *Tohoku Mathematical Journal, Second Series*, 63(4):605–627.

Lin, Y. and Yau, S.-T. (2010). Ricci curvature and eigenvalue estimate on locally finite graphs. *Mathematical research letters*, 17(2):343–356.

Liu, X., Feng, H., Wu, J., and Xia, K. (2021). Persistent spectral hypergraph based machine learning (psh-ml) for protein-ligand binding affinity prediction. *Briefings in Bioinformatics*, 22(5):bbab127.

Newman, M. E. (2006). Modularity and community structure in networks. *Proceedings of the national academy of sciences*, 103(23):8577–8582.

Ni, C.-C., Lin, Y.-Y., Gao, J., Gu, X. D., and Saucan, E. (2015). Ricci curvature of the internet topology. In *2015 IEEE conference on computer communications (INFOCOM)*, pages 2758–2766. IEEE.

Ni, C.-C., Lin, Y.-Y., Luo, F., and Gao, J. (2019). Community detection on networks with ricci flow. *Scientific reports*, 9(1):9984.

Ollivier, Y. (2007). Ricci curvature of metric spaces. *Comptes Rendus Mathematique*, 345(11):643–646.

Samal, A., Sreejith, R., Gu, J., Liu, S., Saucan, E., and Jost, J. (2018). Comparative analysis of two discretizations of ricci curvature for complex networks. *Scientific reports*, 8(1):8650.

Santambrogio, F. (2015). Optimal transport for applied mathematicians. *Birkäuser, NY*, 55(58-63):94.

Sinkhorn, R. (1974). Diagonal equivalence to matrices with prescribed row and column sums. ii. *Proceedings of the American Mathematical Society*, 45(2):195–198.

Sreejith, R., Mohanraj, K., Jost, J., Saucan, E., and Samal, A. (2016). Forman curvature for complex networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2016(6):063206.

Tian, Y., Lubberts, Z., and Weber, M. (2025). Curvature-based clustering on graphs. *Journal of Machine Learning Research*, 26(52):1–67.

Titouan, V., Redko, I., Flamary, R., and Courty, N. (2020). Co-optimal transport. *Advances in neural information processing systems*, 33:17559–17570.

Torres, L., Blevins, A. S., Bassett, D., and Eliassi-Rad, T. (2021). The why, how, and when of representations for complex systems. *SIAM Review*, 63(3):435–485.

Van Der Hoorn, P., Cunningham, W. J., Lippner, G., Trugenberger, C., and Krioukov, D. (2021). Ollivier-ricci curvature convergence in random geometric graphs. *Physical Review Research*, 3(1):013211.

Veličković, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., and Hjelm, R. D. (2018). Deep graph infomax. *arXiv preprint arXiv:1809.10341*.

Yang, C., Wang, R., Yao, S., and Abdelzaher, T. (2022). Semi-supervised hypergraph node classification on hypergraph line expansion. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 2352–2361.

Zhang, J., Gao, M., Yu, J., Guo, L., Li, J., and Yin, H. (2021). Double-scale self-supervised hypergraph learning for group recommendation. In *Proceedings of the 30th ACM international conference on information & knowledge management*, pages 2557–2567.

Zhang, R., Zou, Y., and Ma, J. (2019). Hyper-sagnn: a self-attention based graph neural network for hypergraphs. *arXiv preprint arXiv:1911.02613*.

Zhang, X., Song, D., and Tao, D. (2023). Ricci curvature-based graph sparsification for continual graph representation learning. *IEEE Transactions on Neural Networks and Learning Systems*.

Zhou, Y., Rathore, A., Purvine, E., and Wang, B. (2022). Topological simplifications of hypergraphs. *IEEE Transactions on Visualization and Computer Graphics*, 29(7):3209–3225.

Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., and Wang, L. (2020). Deep graph contrastive representation learning. *arXiv preprint arXiv:2006.04131*.

A Proof of Proposition 1

- We start by computing $w^{(l)}_{\mathcal{N},3}$. Let x and y be two non-gateway nodes from the same clique $(x, y, u_1, \ldots, u_{a-2})$. We have that $St(x) \setminus (x, y) = \{(x, u_1), \ldots, (x, u_{a-2})\}$ and $St(y) \setminus (x, y) = \{(y, u_1), \ldots (y, u_{a-2})\}$. The optimal way to transport the uniform probability measure on $St(x) \setminus (x, y)$ to the uniform probability measure on $St(x) \setminus (x, y)$ is to transport each (x, u_i) to (y, u_i), which are adjacent in the line graph. Each edge has weight $1/(a-2)$ and there is $a-2$ of them, hence a total cost of 1. Hence by induction, since $w^{(l)}_{\mathcal{N},3} = 1$, we have that $w^{(l)}_{\mathcal{N},3} = 1$ for all l.

- For $w^{(l)}_{\mathcal{N},1}$, let x and y be two distinct gateway-nodes, connected by the large hyperedge E_b. We have $St(x) \setminus E_b = \{(x, u_1), \ldots, (x, u_{a-1})\}$ and $St(y) \setminus E_b = \{(y, v_1), \ldots, (x, v_{a-1})\}$. The optimal way to transport the uniform probability measure on $St(x) \setminus E_b$ to the uniform probability measure on $St(y) \setminus E_b$ is to transport each (x, u_i) to a corresponding (y, v_i) via E_b. This corresponds to a path of length 2 on the line graph. Each edge has weight $1/(a-1)$ and there is $a-1$ of them, hence a total cost of 2. As the cost is the same for each pair of nodes in E_b, the flow of E_b is equal to that of any pair $(x, y) \in E_b$. We therefore have that $w^{(l)}_{\mathcal{N},3} = 2$ for all $l \geq 1$.

- For $w^{(l)}_{\mathcal{N},2}$, let x be a gateway node and y and non-gateway node from the same clique $(x, y, u_1, \ldots, u_{a-2}$. We have $St(x) \setminus (x, y) = \{E_b, (x, u_1), \ldots, (x, u_{a-2})\}$ and $St(y) \setminus (x, y) = \{(y, u_1), \ldots, (y, u_{a-2})\}$. We represent the line graph restricted to $St(x) \cup St(y) \setminus (x, y)$ in Figure S.1. We write the mass of each edge according to Equation 3 where we write $d_i = w^{(l-1)}_{\mathcal{N},i}$ to simplify the notation. We have just demonstrated that $d_1 = 2$ and $d_3 = 1$. The optimal transport plan between μ_x and μ_y transports the mass on each (x, u_i) to (y, u_i). These edges are adjacent in the line graph. These $(a-2)$ edges therefore occupy a fraction of the total cost of $\frac{(a-2)d_2}{2+(a-2)d_2}$. The mass $\frac{d_2}{2+(a-2)d_2}$ on E_b is equally transported to each (y, u_i) through the corresponding (x, u_i). Each (y, u_i) is at a distance 2 from E_b, see Figure S.1. This implies a cost of $\frac{2d_2}{2+(a-2)d_2}$ for transporting the mass at E_b. In the end, the total cost of transporting μ_x to μ_y is $\frac{4+(a-2)d_2}{2+(a-2)d_2}$. This implies that

$$w^{(l)}_{\mathcal{N},2} = \frac{4 + (a-2)w^{(l-1)}_{\mathcal{N},2}}{2 + (a-2)w^{(l-1)}_{\mathcal{N},2}}.$$

  We can easily prove by induction on l that for every l, $w^{(l)}_{\mathcal{N},2} \in [1, 2]$.

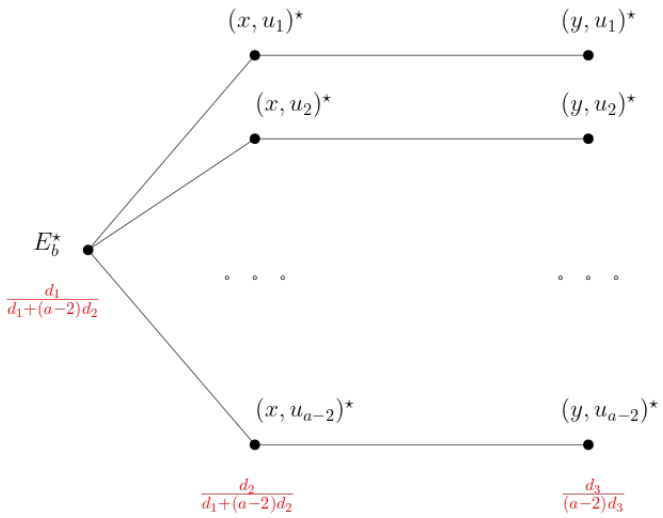


Figure S.1: Restriction of the line graph to compute the flow between a gateway and a non-gateway node in Proposition 1.

B Description of the datasets

The experiments from Section 4.2 have been conducted on these hypergraph datasets:

- Co-citation datasets `Cora-C`, `Citeseer` and `Pubmed` where nodes represent research papers, edges papers co-citing these papers and the communities correspond to similar research thematic.
- The dataset `Cora-A` has a similar structure but is a co-authorship dataset, where hyperedges represent authors.
- In the dataset `Zoo`, nodes are animals, edges represent mutual traits, and communities are types of animals (mammal, fish...).
- In the dataset `Mushroom`, nodes are mushroom samples, edges represent mutual traits, and there are two communities: edible and poisonous.
- In the dataset `NTU2012`, nodes correspond to 3D objects, edges to proximity using image-processing features, and communities are the type of object.

We refer to Table S.1 for a quantitative description of each dataset.

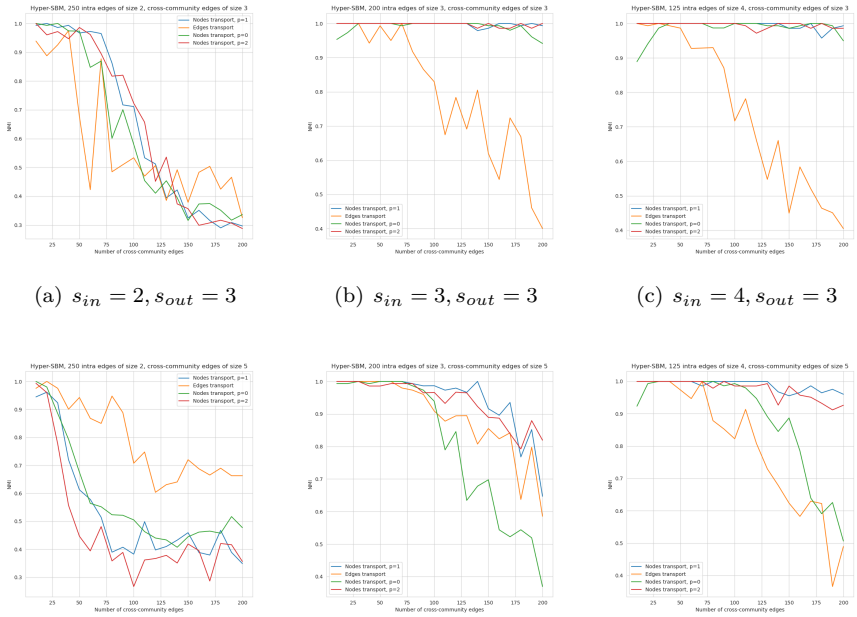
Dataset	Cora-C	Cora-A	Citeseer	Pubmed	Zoo	Mushroom	NTU2012
♯ Nodes	1434	2388	1458	3840	101	8124	2012
♯ Hyperedges	1579	1072	1079	7963	43	298	2012
Avg. hyperedge size	3.0	4.3	3.2	4.4	39.9	136.3	5.0
Avg. node degree	3.3	1.9	2.4	9.0	17.0	5.0	5.0
♯ Communities	7	7	6	3	7	2	67

Table S.1: Basic statistics of hypergraph data used in the experiments.

C Impact of the hyperparameters

Unless specified otherwise, we report the NMI in the synthetic set-up of Section 4.1. Results are averaged over 5 random hypergraph generation and comparison with the ground-truth is realized for the optimal cut-off parameter $\tau^\star$.

Measure parameter p. In Figure S.2, we report the NMI for three values $p = 0, 1, 2$ for the nodes transport, compared with the edge transport. We set Agg to be the maximum function and consider a uniform weighting of the clique graph. We can see that in some cases, there is a clear drop of accuracy for $p = 0$, while $p = 1$ and $p = 2$ display similar performances.

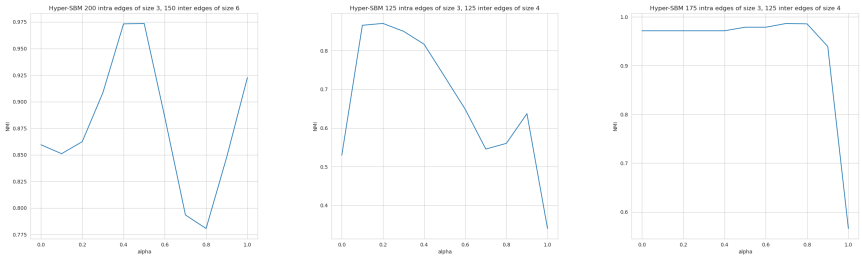


(a) $s_{in} = 2, s_{out} = 3$ (b) $s_{in} = 3, s_{out} = 3$ (c) $s_{in} = 4, s_{out} = 3$

(d) $s_{in} = 2, s_{out} = 5$ (e) $s_{in} = 3, s_{out} = 5$ (f) $s_{in} = 4, s_{out} = 5$

Figure S.2: Study on the impact of the parameter p to cluster hypergraph stochastic block models.

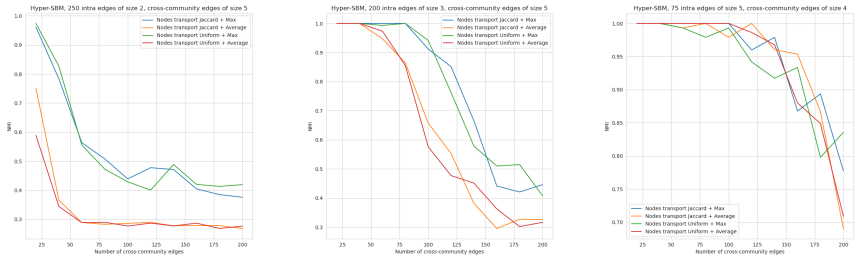
Measure parameter α. In Figure S.3, we report the NMI for a few specific critical values of $s_{in}, s_{out}, N_{in}, N_{out}$. These results are obtained for $p = 1$ for the node transport. We set Agg to be the maximum function and consider a uniform weighting of the clique graph. We can see that in these critical cases, the parameter α can play a very big impact. It is difficult to provide any generic guideline, but extreme values of α seem to provoke instabilities in some cases.



(a) $s_{in} = 3, s_{out} = 6$ (b) $s_{in} = 3, s_{out} = 4$ (c) $s_{in} = 3, s_{out} = 4$

Figure S.3: Study on the impact of the parameter α to cluster hypergraph stochastic block models

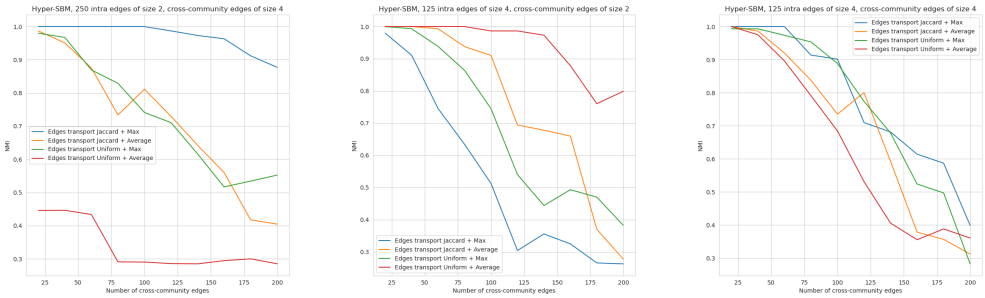
Aggregation and clique graph weighting In Figure S.4 we report the NMI in the synthetic set-up of Section 4.1 where we simultaneously compare the effect of the clique graph weighting and the curvature aggregation at the hyperedge level. We compare the four possible combinations, where the clique graph can have either a Jaccard or a uniform weighting, and Agg can be either the maximum or the arithmetic average. We do not notice any significant difference between the two types of clique graph weightings. However, we observe that using the maximum pairwise curvature instead of the average yields similar to much better results and should therefore always be preferred.



(a) $s_{in} = 2, s_{out} = 5$ (b) $s_{in} = 3, s_{out} = 5$ (c) $s_{in} = 5, s_{out} = 4$

Figure S.4: Study on the impact of the clique graph weighting and the aggregation function to cluster hypergraph stochastic block models with node-Ricci flow

Aggregation and line graph weighting In Figure S.5 we report the NMI in the synthetic set-up of Section 4.1 where we simultaneously compare the effect of the line graph weighting and the curvature aggregation at the hyperedge level. We compare the four possible combinations, where the line graph can have either a Jaccard or a uniform weighting, and Agg can be either the maximum or the arithmetic average. The differences can be very significative, in particular between the combinations Jaccard-maximum and Uniform-average. Overall, we observe that the former provides a much better clustering accuracy whenever $s_{in} \leq s_{out}$ while the latter is better when $s_{in} > s_{out}$. However, we observe in Figure S.6 that in this case, edge-Ricci flow with any parameter still underperforms compared to node-Ricci flow, which is particularly adapted to dealing with the situation of large intra-community hyperedges and small inter-community hyperedges. We consider a node-Ricci flow with parameters Jaccard-maximum. As a result, we claim that the parameter combination Uniform-average for edge-flow should never be preferred as it will always be beaten by either the Jaccard-maximum edge-flow or by the node-Ricci flow.

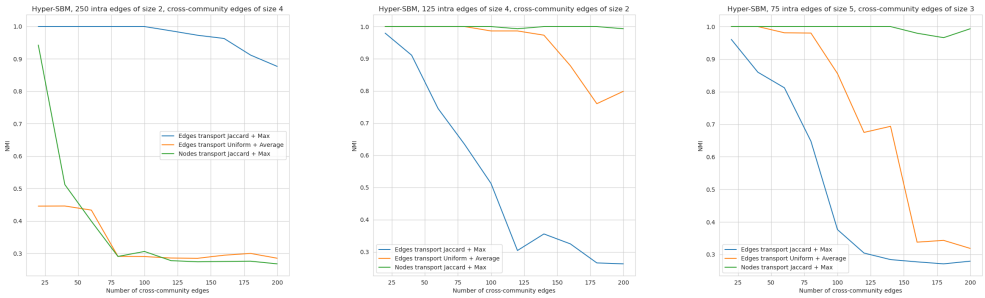


(a) $s_{in} = 2, s_{out} = 4$ (b) $s_{in} = 4, s_{out} = 2$ (c) $s_{in} = 4, s_{out} = 4$

Figure S.5: Study on the impact of the line graph weighting and the aggregation function to cluster hypergraph stochastic block models with edge-Ricci flow

Joint study of every hyperparameter on a real dataset. We study the impact of all parameters except α and τ for the clustering task on the dataset `NTU2012`. The clustering accuracies are reported for the optimal $\tau^\star$. Results are reported in Tables S.2 and S.3. We can see that in this real data case, The choice of the weights for the clique graph and the line graph has a negligible impact. For both types of Ricci flow, considering the maximum curvature over the average one has a tremendous impact. In addition, we can observe an improvement by considering $p = 1$ over $p = 0$. For $p = 2$, numerical instabilities prevented the code from compiling.

Threshold parameter τ We now try to assess the estimation of the $\tau^\star$ parameter providing the highest clustering accuracy. We compare the NMI obtained by trimming edges with a flow greater than $\tau^\star$ with the NMI obtained with τ_H and τ_C. We report the results for the clustering of the datasets `NTU2012` and `Zoo` in Tables S.4 and S.5. Based on the previous findings, we considered a Jaccard weighting, a hyperedge



(a) $s_{in} = 2, s_{out} = 4$ (b) $s_{in} = 4, s_{out} = 2$ (c) $s_{in} = 5, s_{out} = 3$

Figure S.6: Comparison of Jaccard-maximum and Uniform-average edge-Ricci flow with node-Ricci flow.

Aggregation	Clique graph weights	p	NMI
Max	Jaccard	0	79.7
Max	Uniform	0	79.7
Average	Jaccard	0	69.1
Average	Uniform	0	69.1
Max	Jaccard	1	82.1
Max	Uniform	1	82.5
Average	Jaccard	1	69.1
Average	Uniform	1	69.1

Table S.2: Node-Ricci flow clustering of the `NTU2012` dataset for various sets of hyperparameters

Aggregation	Clique graph weights	NMI
Max	Jaccard	80.2
Max	Uniform	80.2
Average	Jaccard	69.1
Average	Uniform	69.1

Table S.3: Edge-Ricci flow clustering of the `NTU2012` dataset for various sets of hyperparameters

aggregation with the max function, and $p = 1$ for the nodes transport. On the `NTU2012` dataset, we observe a better performance using the hypergraph notion of modularity over the clique graph modularity. The loss of accuracy for Edge-Ricci flow is milder than for Node-Ricci flow when using an approximate τ parameter. For the `Zoo` dataset, we observe a very strong difference between the two notions of modularity, as considering the hypergraph modularity yields the optimal clustering, while the modularity on the clique graph gives extremely poor results. This has to be paralleled with the results from Table 1 where methods relying only on the clique graph expansion provided very unsatisfactory results.

Type of flow	$\tau^\star$	τ_H	τ_C
Nodes	82.1	76.3	72.3
Edges	80.2	77.6	75.2

Table S.4: NMI for various choices of τ on the `NTU2012` dataset

Type of flow	$\tau^\star$	τ_H	τ_C
Nodes	96.2	96.2	18.3
Edges	100	100	18.3

Table S.5: NMI for various choices of τ on the `Zoo` dataset