The projection method: a unified formalism for community detection

Martijn Gösgens*, Remco van der Hofstad and Nelly Litvak

Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, Netherlands

We present the class of *projection methods* for community detection that generalizes many popular community detection methods. In this framework, we represent each clustering (partition) by a vector on a high-dimensional *hypersphere*. A community detection method is a projection method if it can be described by the following two-step approach: 1) the graph is mapped to a *query vector* on the hypersphere; and 2) the query vector is projected on the set of clustering vectors. This last *projection* step is performed by minimizing the distance between the query vector and the clustering vector, over the set of clusterings. We prove that optimizing Markov stability, modularity, the likelihood of planted partition models and correlation clustering fit this framework. A consequence of this equivalence is that algorithms for each of these methods can be modified to perform the projection step in our framework. In addition, we show that these different methods suffer from the same *granularity problem*: they have parameters that control the granularity of the resulting clustering, but choosing these to obtain clusterings of the desired granularity is nontrivial. We provide a general heuristic to address this granularity problem, which can be applied to any projection method. Finally, we show how, given a generator of graphs with community structure, we can optimize a projection method for this generator in order to obtain a community detection method that performs well on this generator.

1 Introduction

In complex networks, there often are groups of nodes that are better connected internally than to the rest of the network. In network science, these groups are referred to as *communities*. These communities often have a natural interpretation: they correspond to friend groups in social networks, subject areas in citation networks, or industries in trade networks. Community detection is the task of finding these groups of nodes in a network. This is typically done by partitioning the nodes, so that each node is assigned to exactly one community. There are many different methods for community detection (Fortunato, 2010; Fortunato and Hric, 2016; Rosvall et al., 2019). Yet, it is not easy to say which method is preferable in a given setting.

Most existing methods in network science detect communities by maximizing a *quality function* that measures how well a clustering into communities fits the network at hand. One of the most widely-used quality functions is modularity (Newman and Girvan, 2004), which measures the fraction of edges that are inside communities, and compares this to the

fraction that one would expect in a random graph without community structure. One of the main advantages of modularity maximization is that it does not require one to specify the number of communities that one wishes to detect. However, this does not mean that modularity automatically detects the desired number of communities: it is known that in large networks, modularity maximization is unable to detect communities below or above a given size (Fortunato and Barthélemy, 2007). This problem is often referred to as the *resolution limit*, or the *granularity problem*, of modularity maximization. This problem is often ameliorated by introducing a *resolution parameter* (Reichardt and Bornholdt, 2006; Traag et al., 2011), which allows one to control the range of community sizes that modularity maximization is able to detect.

One of the most popular algorithms for modularity maximization is the *Louvain algorithm* (Blondel et al., 2008). This algorithm performs a greedy maximization and is able to find a local maximum of modularity in linear time (empirically) in the number of network edges.

Community detection is closely related to the more general machine learning task of *data clustering*, as we essentially cluster the nodes based on network topology. In data clustering, the objects to be clustered are typically represented by vectors, and one uses methods like *k*-means (Jain, 2010) or spectral clustering (Von Luxburg, 2007) to find a spatial clustering of these vectors so that nearby vectors are assigned to the same cluster. Community detection can be considered as an instance of clustering, where the elements to be clustered are network nodes.

In this study, we unify several popular community detection and clustering methods into a single geometric framework. We do so by describing a metric space of clusterings, where we represent each clustering C by a binary vector $\boldsymbol{b}(C)$ indexed over the node pairs, i.e., $\boldsymbol{b}(C) \in \mathbb{R}^{\binom{n}{2}}$, where n is the size of the network. We say that a community detection method is a *projection method* if it is equivalent to the following two-step approach: firstly, the graph is mapped to a *query vector* $\boldsymbol{q} \in \mathbb{R}^{\binom{n}{2}}$. Secondly, this query vector is *projected* to the set of clustering vectors. That is, we search for the clustering vector $\boldsymbol{b}(C)$ that minimizes the distance to $\boldsymbol{q}$.

It turns out that many community detection methods fit this framework. In Gösgens et al. (2023a), we prove that modularity maximization is a projection method. In this work, we additionally show that several other popular community detection methods are projection methods. In Section 3, we show that Correlation Clustering (Bansal et al., 2004), the maximization of Markov stability (Delvenne et al., 2010; Lambiotte et al., 2014) and likelihood maximization for several generative models (Avrachenkov et al., 2020) are projection methods. We emphasize that in this paper, we establish equivalences between community detection methods in the strictest mathematical sense. As such, our analytical results are much stronger than merely pointing out that methods are similar or related. Specifically, when we say that two methods are equivalent, we mean that their quality functions f_1 and f_2 define the *exact* same rankings of clusterings, so that for all clusterings C_1, C_2, $f_1(C_1) \geq f_1(C_2)$ holds if and only if $f_2(C_1) \geq f_2(C_2)$.

Some relations between existing community detection methods were already known (Veldt et al., 2018; Newman, 2016). The novelty of this work is that we unify many community detection methods into a single class of projection methods, and uncover the geometric

structure that is baked into each of these methods. Furthermore, we demonstrate the following advantages of this geometric perspective.

Firstly, we show that any community detection method that maximizes or minimizes a weighted sum over pairs of vertices is a projection method. This unifies many well-known methods [Correlation Clustering (Bansal et al., 2004), Markov Stability (Delvenne et al., 2010), modularity maximization (Newman and Girvan, 2004), likelihood maximization (Avrachenkov et al., 2020)], and any other current or future method that can be presented in this form. Importantly, the hyperspherical geometry comes with natural measures for clustering granularity (the latitude) and the similarity between clusterings (the correlation distance). These measures are additionally related to the quality function (the angular distance) by the *hyperspherical law of cosines*, as we explain in Section 2.

Secondly, this geometric framework yields understanding that all community detection methods that are generalized by the projection method, suffer from the same *granularity problem*. That is, these methods require parameter tuning to produce communities of the desired granularity. In Section 5.2 we use the hyperspherical geometry to derive a general heuristic that addresses this problem. We demonstrate that this heuristic, obtained in our earlier work Gösgens et al. (2023b), can be applied to any projection method.

Thirdly, projection methods can be combined by taking linear combinations of their query vectors. In Section 5.3, we demonstrate how we can efficiently find a linear combination that performs well in a given setting.

As a side remark, we note that in network science, the term "clustering" is also used to refer to the abundance of triangles in real-world networks, which is often quantified by the *clustering coefficient* (Watts and Strogatz, 1998; Newman, 2003). The presence of communities usually goes hand in hand with an abundance of triangles (Peixoto, 2022). We emphasize that in the present work, we use the term "clustering" to refer to data clustering, and not to the clustering coefficient. Nevertheless, the global clustering coefficient can be expressed in terms of our hyperspherical geometry (Gösgens et al., 2023a).

1.1 Outline

The remainder of the paper is organized as follows: in Section 2, we describe the projection method and the hyperspherical geometry of clusterings. In Section 3, we prove that several popular community detection methods are projection methods and discuss the implications of these equivalences. In Section 4, we discuss algorithms that can be used to perform the projection step in the projection method. Finally, Section 5 presents methodology for choosing a suitable projection method in given settings. In particular, Section 5.2 discusses how to modify a query mapping in order to obtain a projection method that detects communities of desired granularity, while Section 5.3 demonstrates how we can perform *hyperparameter tuning* within the projection method. Our implementation of the projection method and the experiments of Section 5 is available on Github[1].

---

1 The code is available at https://github.com/MartijnGosgens/hyperspherical_community_detection.

1.2 Notation

This paper discusses the relations between several different community detection methods, that each come with their own notations. We aim to keep notation as consistent as possible, and here we list the most common notation that we will use throughout the paper.

We represent a graph by an $n \times n$ adjacency matrix A with node set $[n] = \{1, \ldots, n\}$ and m edges. We denote the degree of node i by d_i. We write $\sum_{i<j}$ to denote a sum over all node pairs $i, j \in [n]$ with $i < j$. We denote the number of node pairs by $N = \binom{n}{2}$. For a clustering C, we define Intra(C) as the set of intra-cluster node pairs, i.e., pairs of nodes $i < j$ that are part of the same cluster according to C. Similarly, we define Inter(C) as the set of node pairs that are part of different clusters according to C. We define $m_C = |\text{Intra}(C)|$ as the number of intra-cluster pairs. For two functions f, g, we write $f(C) \equiv g(C)$ and say that optimizing f is equivalent to optimizing g if, for each pair of clusterings C_1, C_2, the inequality $f(C_1) \geq f(C_2)$ holds if and only if $g(C_1) \geq g(C_2)$. We denote vectors by bold letters $\boldsymbol{x}, \boldsymbol{y}$ and we denote the inner product between two vectors by $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$. The Euclidean length of a vector is given by $\|\boldsymbol{x}\| = \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle}$.

2 The projection method

In this section, we describe the hyperspherical geometry that the projection method relies on. For more details, we refer to Gösgens et al. (2023b). We consider a graph with n nodes and define a clustering as a partition of these nodes. For a clustering C, we define the *clustering vector* $\boldsymbol{b}(C)$ as the binary vector indexed by the node-pairs, given by

$$\boldsymbol{b}(C)_{ij} = \begin{cases} 1, & \text{if } i \text{ and } j \text{ are in the same cluster,} \\ -1, & \text{if } i \text{ and } j \text{ are in different clusters.} \end{cases}$$

Note that the dimension of this vector is $N = \binom{n}{2}$, and that the Euclidean length of each clustering vector is $\sqrt{N}$, so that they are all located on a hypersphere of radius $\sqrt{N}$ around the origin. Because of this, it is natural to consider the geometry induced by the *angular distance*, given by

$$d_a(\boldsymbol{x}, \boldsymbol{y}) = \arccos\left(\frac{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}{\|\boldsymbol{x}\| \cdot \|\boldsymbol{y}\|}\right). \tag{1}$$

The vector representation $\boldsymbol{b}(C)$, together with the angular distance, defines a hyperspherical geometry of clusterings.

2.1 Clustering granularity and latitude

The clustering into a single community corresponds to the all-one vector $\boldsymbol{b}(C) = \boldsymbol{1}$, while the clustering into n singleton communities corresponds to $\boldsymbol{b}(C) = -\boldsymbol{1}$. These two vectors form opposite *poles* on the hypersphere. The extent to which a clustering resembles the former or latter is referred to as its *granularity*: *fine-grained* clusterings consist of many small communities, while *coarse-grained* clusterings consist of few and large communities. We measure clustering granularity by the *latitude* of the clustering vector. For $\boldsymbol{x} \in \mathbb{R}^N$, the latitude is defined as $\ell(\boldsymbol{x}) = d_a(\boldsymbol{x}, -\boldsymbol{1})$. For a

clustering vector $\boldsymbol{b}(C)$, this is given by $\ell(\boldsymbol{b}(C)) = \arccos(1 - 2\frac{m_C}{N})$, where $m_C = |\text{Intra}(C)|$ is the number of intra-cluster pairs of C. Note that the number of intra-cluster pairs is related to the sum of the cluster sizes: let $s_1, \ldots, s_k$ be the sizes of the clusters of C. Then

$$m_C = \sum_{i=1}^{k} \binom{s_i}{2} = \frac{1}{2} \sum_{i=1}^{k} s_i^2 - \frac{n}{2}.$$

Thus, quantifying clustering granularity by the latitude is equivalent to quantifying it by the sum of squared cluster sizes.

2.2 Parallels and meridians

Borrowing more terminology from geography, for $\lambda \in [0, \pi]$, we define the *parallel* $\mathcal{P}_\lambda$ as the set of vectors with latitude λ. In particular, we refer to $\mathcal{P}_{\pi/2}$ as the *equator*, which corresponds to the set of vectors perpendicular to $\boldsymbol{1}$. For a vector $\boldsymbol{x}$ that is not a multiple of $\boldsymbol{1}$, we define the *parallel projection* $\mathcal{P}_\lambda(\boldsymbol{x})$ as the projection of $\boldsymbol{x}$ onto $\mathcal{P}_\lambda$, and it is given by

$$\mathcal{P}_\lambda(\boldsymbol{x}) = \frac{\sin(\lambda) \cdot \sqrt{N}}{\left\|\boldsymbol{x} - \frac{\langle \boldsymbol{x}, \boldsymbol{1} \rangle}{N}\right\|} \cdot \left(\boldsymbol{x} - \frac{\langle \boldsymbol{x}, \boldsymbol{1} \rangle}{N}\right) - \cos(\lambda) \cdot \boldsymbol{1}. \tag{2}$$

Similarly, we define the *meridian* of $\boldsymbol{x}$ as the one-dimensional line $\{\mathcal{P}_\lambda(\boldsymbol{x}): \lambda \in (0, \pi)\}$.

2.3 Correlation distance and clustering similarity

For three vectors $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{r}$, we can measure the angle on the surface of the hypersphere that the line from $\boldsymbol{x}$ to $\boldsymbol{r}$ makes with the line from $\boldsymbol{y}$ to $\boldsymbol{r}$. This angle $\angle(\boldsymbol{x}, \boldsymbol{r}, \boldsymbol{z})$ is given by the hyperspherical variant of the *law of cosines*:

$$\cos\angle(\boldsymbol{x}, \boldsymbol{r}, \boldsymbol{y}) = \frac{\cos d_a(\boldsymbol{x}, \boldsymbol{y}) - \cos d_a(\boldsymbol{x}, \boldsymbol{r})\cos d_a(\boldsymbol{y}, \boldsymbol{r})}{\sin d_a(\boldsymbol{x}, \boldsymbol{y})\sin d_a(\boldsymbol{y}, \boldsymbol{r})}. \tag{3}$$

In particular, when we take $\boldsymbol{r} = -\boldsymbol{1}$, this angle corresponds to the angle between the meridians of $\boldsymbol{x}$ and $\boldsymbol{y}$. This angle turns out to have an interesting interpretation, as stated in Theorem 1:

Theorem 1. (Gösgens et al., 2023a). *For two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ that are not multiples of $\boldsymbol{1}$, the angle that their meridians make is equal to the arccosine of the Pearson correlation between $\boldsymbol{x}$ and $\boldsymbol{y}$.*

Because of Theorem 1, we call this angle the *correlation distance* between $\boldsymbol{x}$ and $\boldsymbol{y}$. Note that $d_a(\boldsymbol{x}, -\boldsymbol{1}) = \ell(\boldsymbol{x})$, so that the correlation distance is given by

$$d_\rho(\boldsymbol{x}, \boldsymbol{y}) = \arccos\left(\frac{\cos d_a(\boldsymbol{x}, \boldsymbol{y}) - \cos \ell(\boldsymbol{x})\cos \ell(\boldsymbol{y})}{\sin \ell(\boldsymbol{x})\sin \ell(\boldsymbol{y})}\right), \tag{4}$$

and the Pearson correlation between vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ is thus given by $\cos d_\rho(\boldsymbol{x}, \boldsymbol{y})$. The correlation coefficient between two clustering vectors $\boldsymbol{b}(C), \boldsymbol{b}(T)$ turns out to be a useful quantity for measuring the similarity between clusterings C and T (Gösgens et al., 2021). There exist many measures to quantify the similarity between two clusterings, but most of these measures suffer from the defect that they are biased towards either coarse- or fine-grained

clusterings (Vinh et al., 2009; Lei et al., 2017). The Pearson correlation between two clustering vectors does not suffer from this bias, and additionally satisfies many other desirable properties (Gösgens et al., 2021). Because of that, we will use the Pearson correlation to measure the similarity between clusterings. For clusterings C and T, the Pearson correlation is given by

$$\rho(C, T) = \cos d_\rho(\boldsymbol{b}(C), \boldsymbol{b}(T)) = \frac{m_{CT} \cdot N - m_C \cdot m_T}{\sqrt{m_C \cdot (N - m_C) \cdot m_T \cdot (N - m_T)}}, \tag{5}$$

where $m_C = |\text{Intra}(C)|$, $m_T = |\text{Intra}(T)|$ and $m_{CT} = |\text{Intra}(C) \cap \text{Intra}(T)|$. In cases where C and T correspond to the detected and planted (i.e., ground-truth) clusterings, we use $\rho(C, T)$ as a measure of the performance of the community detection.

2.4 Query mappings

Above, we have defined the hyperspherical geometry of clusterings. This geometry comes with natural measures for clustering granularity (the latitude ℓ) and similarity between clusterings (the correlation ρ). The idea behind the projection method is that we map a graph to a point in this same geometry, and then find the clustering vector that is closest to that point. For a graph with adjacency matrix A, we denote the vector that it is mapped to by $\boldsymbol{q}(A) \in \mathbb{R}^N$, and refer to it as the *query vector* of A. We refer to $\boldsymbol{q}(\cdot)$ as the *query mapping*. The name 'query' comes from the fact that in the second step of the projection method, we search for the clustering vector $\boldsymbol{b}(C)$ that minimizes $d_a(\boldsymbol{q}(A), \boldsymbol{b}(C))$. That is, among the set of clustering vectors, we find the one that is *nearest to* $\boldsymbol{q}(A)$. In short, we arrive at the following definition of the projection method:

Definition 1. *A community detection method is a projection method if it can be described by the following two-step approach: (1) the graph with adjacency matrix A is first mapped to a query vector $\boldsymbol{q}(A)$; and (2) the query vector is projected to the set of clustering vectors by minimizing $d_a(\boldsymbol{q}(A), \boldsymbol{b}(C))$ over the set of clusterings C.*

There exist infinitely many ways to map graphs to query vectors. One of the simplest ways is to simply turn the adjacency matrix into a vector like $\boldsymbol{q}(A)_{ij} = \frac{1}{2}(A_{ij} + A_{ji})$, where the average is taken in case A is directed. In general, we define the *half-vectorization* of a matrix $X \in \mathbb{R}^{n \times n}$ by

$$\boldsymbol{v}(X)_{ij} = \frac{1}{2}(X_{ij} + X_{ji}).$$

The vector $\boldsymbol{v}(A^r)$ counts the number of paths of length r between each pair of vertices. In particular, $\boldsymbol{v}(A^2)_{ij}$ corresponds to the number of neighbors that the nodes i and j share.

There are many more ways in which we can construct a query vector based on A. For example, the entry $\boldsymbol{q}(A)_{ij}$ can also depend on the degrees of i and j or even the length of the shortest path between i and j.

Finally, because we are minimizing the angular distance, the length of $\boldsymbol{q}(A)$ is not relevant. It may be natural to normalize all vectors to a Euclidean length $\sqrt{N}$ so that they have the same length as clustering vectors, but we will not do so to avoid cluttering the notation.

In summary, the hyperspherical geometry comes with three key measures: firstly, the *angular distance* $d_a(\boldsymbol{q}(A), \boldsymbol{b}(C))$ is the quality measure that we minimize in order to detect communities. Secondly, the *latitude* measures the granularity of a clustering. That is, $\ell(\boldsymbol{b}(T))$ measures the granularity of the planted clustering, while $\ell(\boldsymbol{b}(C))$ measures the granularity of the detected clustering. Thirdly, the *correlation distance* between the planted and detected communities $d_\rho(\boldsymbol{b}(C), \boldsymbol{b}(T))$ [or its cosine, $\rho(C, T) = \cos d_\rho(\boldsymbol{b}(C), \boldsymbol{b}(T))$] measures the performance of the detection. In Section 5.2, we will additionally see that $d_\rho(\boldsymbol{q}(A), \boldsymbol{b}(T))$ is a useful measure. The angular distance, correlation distance and latitude are related by the *hyperspherical law of cosines*, given in (4).

3 Equivalences to other community detection methods

In this section, we will prove that the class of projection methods generalizes several community detection methods. We prove that the class of projection methods is equivalent to correlation clustering, and we prove that the remaining methods are subclasses of the class of projection methods. For each of the related clustering and community detection methods, we will provide the query mapping of the corresponding projection method. The relations between the methods that are discussed here are illustrated in Figure 1.

3.1 Correlation clustering

Correlation clustering (Bansal et al., 2004) is a framework for clustering where pairwise similarity and dissimilarity values w_{ij}^+ and w_{ij}^- are given for every pair of objects i, j. The objective is to maximize the similarity within the clusters and the dissimilarity between the clusters. Such *agreement* of a clustering C with weights $w_{ij}^\pm$ is expressed as

$$\text{CorClust}_{\max}(C; w^+, w^-) = \sum_{ij \in \text{Intra}(C)} w_{ij}^+ + \sum_{ij \in \text{Inter}(C)} w_{ij}^-.$$

Correlation clustering solves the maximization problem

$$\max_C \text{CorClust}_{\max}(C; w^+, w^-). \tag{6}$$

Equivalently, one can express the *disagreement* of a clustering C with weights $w_{ij}^\pm$ as

$$\begin{aligned} &\text{CorClust}_{\min}(\mathcal{C}(w^+, w^-); w^+, w^-) \\ &= \sum_{ij \in \text{Intra}(C)} w_{ij}^- + \sum_{ij \in \text{Inter}(C)} w_{ij}^+ \\ &= \sum_{i<j}(w_{ij}^+ + w_{ij}^-) - \text{CorClust}_{\max}(\mathcal{C}(w^+, w^-); w^+, w^-). \end{aligned}$$

Then (6) can be stated as a minimization problem

$$\min_C \text{CorClust}_{\min}(C; w^+, w^-). \tag{7}$$

Somewhat counter-intuitively, the equivalent formulations (6) and (7) lead to different approximation results. Indeed, for minimization problems (7), an α-approximation guarantee means
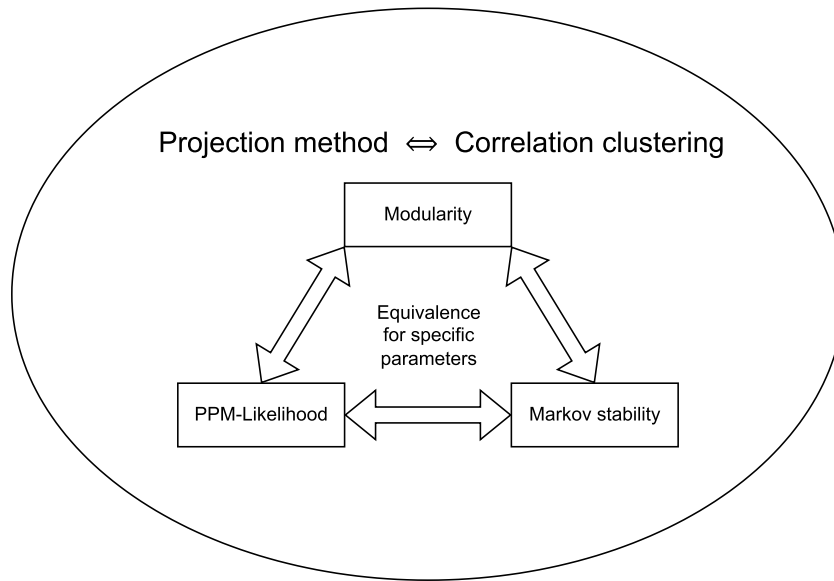
FIGURE 1
A schematic overview of the equivalences between the community detection and clustering methods that are described in Section 3. The projection method is completely equivalent to correlation clustering, while modularity, PPM-likelihood and Markov stability are subsets. For certain parameter choices, these latter three methods are equivalent.

that for a given $\alpha > 1$, an optimization algorithm $\mathcal{C}$ satisfies the condition

$$\text{CorClust}_{\min}\left(\mathcal{C}\left(w^+, w^-\right); w^+, w^-\right)$$
$$\leq \alpha \cdot \text{CorClust}_{\min}\left(C^\star; w^+, w^-\right), \quad \text{for any } w^+, w^-, \quad (8)$$

where $C^\star$ solves (7). On the other hand, for the maximization problem (6), a β-approximation guarantee means that for a given $\beta < 1$, the optimization algorithm $\mathcal{C}$ satisfies the condition

$$\text{CorClust}_{\max}\left(\mathcal{C}\left(w^+, w^-\right); w^+, w^-\right)$$
$$\geq \beta \cdot \text{CorClust}_{\min}\left(C^\star; w^+, w^-\right), \quad \text{for any } w^+, w^-. \quad (9)$$

Now, suppose, we have found an α-approximation in (8). Then

$$\text{CorClust}_{\max}\left(\mathcal{C}\left(w^+, w^-\right); w^+, w^-\right)$$
$$= \sum_{i<j}\left(w_{ij}^+ + w_{ij}^-\right) - \text{CorClust}_{\min}\left(\mathcal{C}\left(w^+, w^-\right); w^+, w^-\right)$$
$$\geq \sum_{i<j}\left(w_{ij}^+ + w_{ij}^-\right) - \alpha \cdot \text{CorClust}_{\min}\left(C^\star; w^+, w^-\right)$$
$$= \alpha \cdot \text{CorClust}_{\max}\left(C^\star; w^+, w^-\right) - (\alpha - 1) \cdot \sum_{i<j}\left(w_{ij}^+ + w_{ij}^-\right),$$

which gives *some* approximation guarantee for the maximization problem, but not of the same multiplicative form as (9).

The motivation for correlation clustering originates from the setting where we are given a noisy classifier that, for each pair of objects, predicts whether they should be clustered together or apart (Bansal et al., 2004). This leads to the simple ±1 version of correlation clustering, where $w_{ij}^+, w_{ij}^- \in \{0, 1\}$ and $w_{ij}^+ + w_{ij}^- = 1$ (or equivalently, $w_{ij}^+ - w_{ij}^- = \pm 1$) holds for each pair ij. The best known approximation guarantee for the minimization formulation of the ±1 variant is 2.06, which is achieved by rounding the solution from an LP relaxation (Chawla et al., 2015).

For the general case where the weights are unconstrained, it is known that maximizing agreement is APX-hard (Charikar et al., 2005), which means that any constant-factor approximation is NP-hard. The same authors do provide a 0.7666-approximation algorithm by rounding the semi-definite programming solution. Note that this does not contradict the APX-hardness result, as semi-definite programming is also NP-hard.

We now prove that correlation clustering corresponds to a projection method:

Lemma 1. *Correlation clustering with similarity and dissimilarity values* $w^+ = (w_{ij}^+)_{i<j}, w^- = (w_{ij}^-)_{i<j}$ *is equivalent to a projection method with query vector* $\mathbf{q}^{(\text{CC})}$ *given by*

$$\mathbf{q}_{ij}^{(\text{CC})} = w_{ij}^+ - w_{ij}^-.$$

Proof. Minimizing $d_a\left(\mathbf{q}^{(\text{CC})}, \mathbf{b}(C)\right)$ is equivalent to maximizing $\langle \mathbf{q}^{(\text{CC})}, \mathbf{b}(C) \rangle = \|\mathbf{q}^{(\text{CC})}\| \cdot \sqrt{N} \cdot \arccos d_a\left(\mathbf{q}^{(\text{CC})}, \mathbf{b}(C)\right)$. We prove that this, in turn, is equivalent to maximizing $\text{CorClust}_{\max}$ $(C; w^+, w^-)$ w.r.t. to the given values $(w_{ij}^+)_{i<j}$ and $(w_{ij}^-)_{i<j}$:

$$\langle \mathbf{q}^{(\text{CC})}, \mathbf{b}(C) \rangle = \sum_{ij \in \text{Intra}(C)}\left(w_{ij}^+ - w_{ij}^-\right) + \sum_{ij \in \text{Inter}(C)}\left(w_{ij}^- - w_{ij}^+\right)$$
$$= 2 \cdot \left(\sum_{ij \in \text{Intra}(C)} w_{ij}^+ + \sum_{ij \in \text{Inter}(C)} w_{ij}^-\right) - \sum_{i<j}\left(w_{ij}^+ + w_{ij}^-\right)$$
$$= 2 \cdot \text{CorClust}_{\max}\left(C; w^+, w^-\right) - \sum_{i<j}\left(w_{ij}^+ + w_{ij}^-\right).$$

The last term does not depend on C, so we obtain that indeed maximizing $\langle \mathbf{q}^{(\text{CC})}, \mathbf{b}(C) \rangle$ is equivalent to maximizing $\text{CorClust}_{\max}$ $(C; w^+, w^-)$, as required. □

It is easy to see that any query vector $\boldsymbol{q}$ can also be turned into a correlation clustering objective by taking $w_{ij}^+ = \max\{0, \boldsymbol{q}_{ij}\}$ and $w_{ij}^- = \max\{0, -\boldsymbol{q}_{ij}\}$. This tells us that correlation clustering and projection methods are *equivalent*, in the sense that any correlation clustering instance can be mapped to a query vector and *vice versa*. Note, however, that both classes come with different invariances: correlation clustering is invariant to applying the same linearly increasing transformation $f(x) = a + bx$ to all values w_{ij}^+ and w_{ij}^-, where $a, b \in \mathbb{R}$ and $b > 0$. Similarly, projection methods are invariant to multiplying the query vector by any positive constant, due to the hyperspherical geometry.

The equivalence between correlation clustering and projection methods gives a new interpretation for correlation clustering in terms of hyperspherical geometry, and allows one to transfer hardness results from correlation clustering to projection methods.

Correlation clustering vs. community detection

While the fields of community detection and correlation clustering are similar, the focus of the two fields is notably different: correlation clustering studies clustering from an algorithmic viewpoint, where the goal is to design algorithms with provable optimization guarantees with respect to the correlation clustering quality function. In contrast, community detection focuses mainly on the choice of the quality function, with the aim to obtain a meaningful clustering into communities. In this context, 'meaningful' can mean either statistically significant, or similar to some ground truth clustering. In summary, community detection asks "What quality function to optimize?" while correlation clustering asks "What algorithm is best for optimizing the correlation clustering quality function?."

Veldt et al. (2018) introduced an interesting variant of correlation clustering methods known as *LambdaCC*, and showed that it is related to Sparsest Cut, Normalized Cut and Cluster Deletion. They additionally introduce a degree-corrected variant of LambdaCC and prove that it is equivalent to CL-modularity, which we discuss in Section 3.3.

3.2 Markov stability

The *Markov stability* (Delvenne et al., 2010) of a clustering with respect to a network quantifies how likely a random walker is to find itself in the same community at the beginning and end of some time interval. When communities are clearly present in a network, then a random walker will tend to stay inside communities for long time periods and travel between communities infrequently. Markov stability can be defined for various types of discrete- or continuous-time random walks (Lambiotte et al., 2014). Let $P(t)_{ij}$ be the probability that the random walker is at node j at time t if it was at node i at time 0, and denote by $P(t) = (P(t)_{ij}) \in \mathbb{R}^{n \times n}$ the t-transition matrix of the random walk. For simplicity, in this paper, we only consider discrete time $t = 0, 1, \ldots$, thus $P(t) = P^t$, where $P = P(1)$. We assume that P is irreducible and aperiodic, so that the random walk has a unique stationary distribution $\boldsymbol{s} \in \mathbb{R}^n$, which is the unique solution to $\boldsymbol{s} = \boldsymbol{s}P$, such that all elements of $\boldsymbol{s}$ sum up to one. We consider a random walker starting from the initial state sampled from $\boldsymbol{s}$, and compare the distribution of its location at time t, to another location sampled from the stationary distribution. Markov

stability measures the covariance between the community label indicators before and after the interval t.

Formally, let C be a clustering and let the clusters be numbered 1, 2, $\ldots, k$, where k is the number of clusters in C. We denote by $H(C)$ the $n \times k$ indicator matrix of the clustering C, where $H(C)_{ia} = 1$ if node i belongs to the a-th cluster, and $H(C)_{ia} = 0$ otherwise. Markov stability is defined as

$$\text{MarkovStability}(C, P(t)) = \text{Trace}\left(H(C)^\top \left(\text{diag}(\boldsymbol{s})P(t) - \boldsymbol{s}^\top \boldsymbol{s}\right)H(C)\right). \quad (10)$$

For more details on the definition of Markov stability and its variants, we refer to Delvenne et al. (2010) and Lambiotte and Schaub (2021). We show that maximizing Markov stability is a projection method with respect to the query vector

$$\boldsymbol{q}_t^{(\text{MS})} = \boldsymbol{v}\left(\text{diag}(\boldsymbol{s})P(t) - \boldsymbol{s}^\top \boldsymbol{s}\right), \quad (11)$$

where $\boldsymbol{v}(X) \in \mathbb{R}^N$ is the *half-vectorization* of the matrix X, defined by $\boldsymbol{v}(X)_{ij} = \frac{1}{2}(X_{ij} + X_{ji})$ for $i < j$. We show that for *any* matrix $X \in \mathbb{R}^{n \times n}$, maximizing $\text{Trace}\,(H(C)^\top X H(C))$ is a projection method:

Lemma 2. *For any matrix $X \in \mathbb{R}^{n \times n}$, maximizing the trace*

$$Trace\left(H(C)^\top X H(C)\right) \quad (12)$$

over the set of all clusterings C is equivalent to minimizing $d_a(\boldsymbol{v}(X), \boldsymbol{b}(C))$.

Proof. The trace is written as

$$\text{Trace}\left(H(C)^\top X H(C)\right) = \sum_a \left[H(C)^\top X H(C)\right]_{aa}.$$

We write

$$\left[H(C)^\top X H(C)\right]_{aa} = \sum_i \sum_j H(C)_{ia} X_{ij} H(C)_{ja}. \quad (13)$$

Now, note that $H(C)_{ia}H(C)_{ja} = 1$ whenever i and j are both in community a, and $H(C)_{ia}H(C)_{ja} = 0$ otherwise. Therefore, summing over a, we get $\sum_a H(C)_{ia}H(C)_{ja} = 1$ if $ij \in \text{Intra}(C)$, $ji \in \text{Intra}(C)$ or $i = j$. Hence,

$$\begin{aligned}\text{Trace}\left(H(C)^\top X H(C)\right) &= \sum_a \left[H(C)^\top X H(C)\right]_{aa} \\ &= \sum_{ij \in \text{Intra}(C)} \left(X_{ij} + X_{ji}\right) + \sum_{i \in [n]} X_{ii}.\end{aligned}$$

Note that the sum $\sum_{i \in [n]} X_{ii}$ does not depend on C, so that omitting it will not affect the optimization. In addition, we can subtract $\frac{1}{2}\sum_{i<j}(X_{ij} + X_{ji})$, which is also constant w.r.t. C. We obtain

$$\begin{aligned}\text{Trace}\left(H(C)^\top X H(C)\right) &\equiv \sum_{ij \in \text{Intra}(C)} \frac{1}{2}\left(X_{ij} + X_{ji}\right) \\ &\quad - \sum_{ij \in \text{Inter}(C)} \frac{1}{2}\left(X_{ij} + X_{ji}\right) \\ &= \langle \boldsymbol{v}(X), \boldsymbol{b}(C)\rangle.\end{aligned}$$

To conclude, trace maximization is equivalent to maximizing $\langle \boldsymbol{v}(X), \boldsymbol{b}(C)\rangle$, which is equivalent to minimizing $d_a(\boldsymbol{v}(X), \boldsymbol{b}(C))$ over the set of clusterings C. □

It is known (Delvenne et al., 2010) that for a discrete-time Markov chain and $t = 1$, Markov stability is equivalent to CL-modularity

maximization with $\gamma = 1$, which we define in Section 3.3. The time parameter t controls the granularity of the detected communities. When $t = 0$, we get communities of size 1, because $P^0 = I$, so that $(\mathrm{diag}(\boldsymbol{s})P^t - \boldsymbol{s}^\top \boldsymbol{s})_{ij} = -s_i s_j < 0$ for all $i \neq j$, therefore (10) is maximized when in (13) we have $H_{ia}H_{ja} = 0$ for all a. Furthermore, Delvenne et al. (2010) show that in the limit $t \to \infty$, Markov stability in continuous time and with normalized Laplacian instead of the matrix P, divides the network in two communities, corresponding to the positive and the negative coordinates of the *Fiedler vector*, that is the eigenvector corresponding to the second smallest eigenvalue of the normalized Laplacian.

Matrix vs. vector representation

Note that the vector and matrix representations of clusterings are related by $\boldsymbol{b}(C) = 2\boldsymbol{v}\left(H(C)H(C)^\top\right) - \boldsymbol{1}$. This relation makes it possible to re-define the hyperspherical geometry from Section 2 entirely in terms of $n \times n$ matrices instead of $\binom{n}{2}$-dimensional vectors. However, we refrain from doing so, because we believe that in most cases, the vector representations are easier to work with. To illustrate, the matrix formulation of projection methods amounts to replacing the *query vector* $\boldsymbol{q}$ with a *query matrix* $Q \in \mathbb{R}^{n \times n}$. Note that Q has n^2 entries $i, j \in [n]$, while $\boldsymbol{q}$ has only $\binom{n}{2}$ entries $i < j$, but these extra entries do not carry any additional information. Indeed, concerning the off-diagonal elements $i > j$, it does not matter whether Q is symmetric or not, because $\mathrm{Trace}(H(C)^\top Q H(C)) = \frac{1}{2}\mathrm{Trace}(H(C)^\top (Q + Q^\top)H(C))$ for any $Q \in \mathbb{R}^{n \times n}$. Furthermore, the diagonal entries $i = j$ play no role because adding any diagonal matrix $\mathrm{diag}(\boldsymbol{y})$, for $\boldsymbol{y} \in \mathbb{R}^n$, to Q, does not affect the optimization:

$$
\begin{aligned}
&\mathrm{Trace}\left(H(C)^\top \left(Q + \mathrm{diag}(\boldsymbol{y})\right)H(C)\right) \\
&= \mathrm{Trace}\left(H(C)^\top Q H(C)\right) + \sum_{i \in [n]} \boldsymbol{y}_i \equiv \mathrm{Trace}\left(H(C)^\top Q H(C)\right).
\end{aligned}
$$

The vector representation has the advantage that it omits these unimportant values, which is why we use query vectors instead of query matrices. Nevertheless, the matrix representation allows for an easier analysis in some settings. For example, in Liu and Barahona (2018), the spectral properties of Markov stability are leveraged to create an optimization algorithm.

3.3 Modularity

Modularity maximization is one of the most widely-used community detection methods (Newman and Girvan, 2004). Modularity measures the excess of edges inside communities, compared to a null model; a random graph model without community structure. This null model is usually either the Erdős-Rényi (ER) model or the Chung-Lu [CL, Chung and Lu (2001)] model. Modularity comes with a *resolution parameter* that controls the granularity of the detected clustering. For the ER and CL null models, modularity is given by

$$
\mathbf{CLM}(C; A, \gamma) = \frac{1}{2m} \sum_{ij \in \mathrm{Intra}(C)} \left(A_{ij} - \gamma \frac{d_i d_j}{2m} \right),
$$

$$
\mathbf{ERM}(C; A, \gamma) = \frac{1}{2m} \sum_{ij \in \mathrm{Intra}(C)} A_{ij} - \gamma \frac{m}{N},
$$

FIGURE 2
Illustration of the geodesics formed by varying the resolution parameter of the modularity vector for a fixed null model. The ER-modularity vectors lie on a single meridian, in contrast to the CL-modularity vectors.
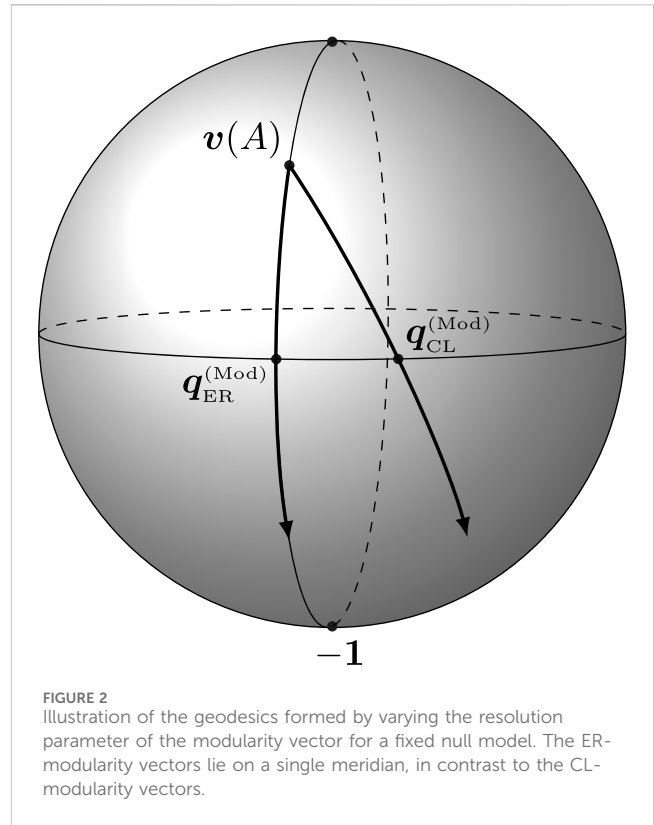
where we recall that d_i is the degree of node i, $m = \frac{1}{2}\sum_{i \in [n]} d_i$ is the number of edges, and $N = \binom{n}{2}$ is the number of node pairs.

In Gösgens et al. (2023b), we have proven that modularity maximization is a projection method. In addition, the equivalence between modularity maximization and correlation clustering was already proven by Veldt et al. (2018). Hence, Lemma 1, too, establishes that modularity maximization is a projection method. Finally, Newman (2006) shows that modularity can be written in a similar trace-maximization form as (12), which additionally allows one to use Lemma 2 to prove that modularity maximization is a projection method. Either way, we get the following query vectors:

$$
\boldsymbol{q}_{\mathrm{CL}}^{(\mathrm{Mod})}(A, \gamma) = \boldsymbol{v}(A) - \gamma \cdot \boldsymbol{d}(A), \quad \boldsymbol{q}_{\mathrm{ER}}^{(\mathrm{Mod})}(A, \gamma) = \boldsymbol{v}(A) - \gamma \frac{m}{N}\boldsymbol{1},
$$

where $\boldsymbol{v}(A)$ is the *adjacency vector* (the half-vectorization of the adjacency matrix); m is the number of edges; and $\boldsymbol{d}(A)_{ij} = \frac{1}{2m}d_i d_j$. For a fixed null model, the vectors that are obtained by varying γ have the following geometric interpretation (Gösgens et al., 2023a): because the query vector $\boldsymbol{q}_{\mathrm{ER}}^{(\mathrm{Mod})}$ is a linear combination of $\boldsymbol{v}(A)$ and $\boldsymbol{1}$ with coefficients depending on γ, it can be shown that $\boldsymbol{q}_{\mathrm{ER}}^{(\mathrm{Mod})}(A, \gamma)$ lies on the meridian of $\boldsymbol{v}(A)$ for every value of γ. The latitude of this modularity vector is related to the resolution parameter by

$$
\tan \ell\left(\boldsymbol{q}_{\mathrm{ER}}^{(\mathrm{Mod})}(A, \gamma)\right) = \frac{\sqrt{\frac{N-m}{m}}}{\gamma - 1}. \tag{14}
$$

The vector $\boldsymbol{q}_{\mathrm{CL}}^{(\mathrm{Mod})}(A, \gamma)$ does not lie on a single meridian and its latitude does not allow for such an elegant formula. However, the set

$(\boldsymbol{q}_{\mathrm{CL}}^{(\mathrm{Mod})}(A, \gamma))_{\gamma \geq 0}$ does correspond to a *geodesic* on the hypersphere, which is the spherical analogue to a straight line in Euclidean geometry. This geodesic runs from $\boldsymbol{v}(A)$ to $-\boldsymbol{d}(A)$. In general, let $\mathcal{N}$ be a null model and let $\boldsymbol{p}^{\mathcal{N}}(A)$ be the corresponding vector of expected edge probabilities for the null model $\mathcal{N}$, then the set $(\boldsymbol{q}_{\mathcal{N}}^{(\mathrm{Mod})}(A, \gamma))_{\gamma \in [0, \infty]}$ corresponds to the geodesic between $\boldsymbol{v}(A)$ and $-\boldsymbol{p}^{\mathcal{N}}(A)$. In Figure 2, we illustrate the geodesics formed by the modularity vectors for the ER and CL null models.

The granularity problem of modularity maximization

It is well-known that modularity maximization is unable to detect communities that are either too large or too small (Fortunato and Barthélemy, 2007). This behavior is often demonstrated by the *ring of cliques*, a graph consisting of k cliques of size s each, where each clique is connected to the next one by a single edge. Let us denote the adjacency matrix of this ring of cliques by $A_{k,s}$, and let $T_{k,s}$ denote its natural clustering into k cliques. For fixed γ and s and sufficiently large k, modularity-maximizing algorithms will merge neighboring cliques. Geometrically, this problem can be understood as follows: because $N = \Theta(k^2)$ and $m = \Theta(k)$, (14) tells us that $\tan \ell(\boldsymbol{q}_{\mathrm{ER}}^{(\mathrm{Mod})}(A_{k,s}, \gamma)) = \Theta(\sqrt{k})$ if $\gamma \neq 1$ and $\tan \ell(\boldsymbol{q}_{\mathrm{ER}}^{(\mathrm{Mod})}(A_{k,s}, \gamma)) = +\infty$ if $\gamma = 1$. Thus, for any fixed $\gamma > 0$, we have $\ell(\boldsymbol{q}_{\mathrm{ER}}^{(\mathrm{Mod})}(A_{k,s}, \gamma)) \to \frac{\pi}{2}$ as $k \to \infty$. However, the clustering $T_{k,s}$ has $|\mathrm{Intra}(T_{k,s})| = \Theta(k)$, so that $\ell(\boldsymbol{b}(T_{k,s})) = \arccos(1 - 2 \cdot |\mathrm{Intra}(T_{k,s})|/N) \to \arccos(1) = 0$. Now, by the inverse triangle inequality, we have

$$d_a\big(\boldsymbol{q}_{\mathrm{ER}}^{(\mathrm{Mod})}, \boldsymbol{b}(T_{k,s})\big) \geq |d_a\big(\boldsymbol{q}_{\mathrm{ER}}^{(\mathrm{Mod})}, -\boldsymbol{1}\big) - d_a(-\boldsymbol{1}, \boldsymbol{b}(T_{k,s}))|$$
$$= |\ell\big(\boldsymbol{q}_{\mathrm{ER}}^{(\mathrm{Mod})}\big) - \ell(\boldsymbol{b}(T_{k,s}))| \to \frac{\pi}{2}.$$

Note that half of the hypersphere lies within a distance $\frac{\pi}{2}$ from the vector $\boldsymbol{q}_{\mathrm{ER}}^{(\mathrm{Mod})}$. And indeed, it can be shown that the clustering consisting of pairs of neighboring cliques is closer to $\boldsymbol{q}_{\mathrm{ER}}^{(\mathrm{Mod})}$ than $\boldsymbol{b}(T_{k,s})$, so that modularity-maximizing algorithms will prefer such clusterings over the natural clustering into cliques.

Note that by (14), the latitude of the ER-modularity vector is a monotone function of the resolution parameter γ. Thus, choosing the resolution parameter is equivalent to setting the latitude of the query vector $\lambda = \ell(\boldsymbol{q}_{\mathrm{ER}}^{(\mathrm{Mod})})$. The hyperspherical geometry suggests two ways to choose λ. The first approach is to choose λ to minimize the angular distance $d_a(\boldsymbol{q}_{\mathrm{ER}}^{(\mathrm{Mod})}, \boldsymbol{b}(T_{k,s}))$. Note that changing the query latitude λ does not affect the correlation distance $d_\rho(\boldsymbol{q}_{\mathrm{ER}}^{(\mathrm{Mod})}, \boldsymbol{b}(T_{k,s}))$. This allows us to use the law of cosines (3), to express $\cos d_a(\boldsymbol{q}_{\mathrm{ER}}^{(\mathrm{Mod})}, \boldsymbol{b}(T_{k,s}))$ as a function of λ, $\lambda_T = \ell(\boldsymbol{b}(T_{k,s}))$, and $\theta = d_\rho(\boldsymbol{q}_{\mathrm{ER}}^{(\mathrm{Mod})}, \boldsymbol{b}(T_{k,s}))$ as

$$\cos d_a\big(\boldsymbol{q}_{\mathrm{ER}}^{(\mathrm{Mod})}, \boldsymbol{b}(T_{k,s})\big) = \cos \lambda \cdot \cos \lambda_T + \cos \theta \cdot \sin \lambda \cdot \sin \lambda_T.$$

Minimizing $d_a(\boldsymbol{q}_{\mathrm{ER}}^{(\mathrm{Mod})}, \boldsymbol{b}(T_{k,s}))$ as a function of λ yields $\tan \lambda = \cos \theta \tan \lambda_T$, i.e., $\lambda \sim \cos(\theta)\lambda_T$ as $\lambda_T \to 0$. The second approach for choosing λ is to simply equate the query latitude to the latitude of $T_{k,s}$, so $\lambda = \lambda_T$. Both approaches yield $\lambda \to 0$ as $k \to \infty$, so that by the triangle inequality the distance between $T_{k,s}$ and the modularity vector with latitude λ vanishes as

$$d_a\big(\boldsymbol{q}_{\mathrm{ER}}^{(\mathrm{Mod})}, \boldsymbol{b}(T_{k,s})\big) \leq d_a\big(\boldsymbol{q}_{\mathrm{ER}}^{(\mathrm{Mod})}, -\boldsymbol{1}\big) + d_a(-\boldsymbol{1}, \boldsymbol{b}(T_{k,s})) = \lambda + \lambda_T \to 0.$$

In terms of the resolution parameter γ, both these approaches yield $\gamma = \Theta(k)$. Moreover, it can be shown that for both of these approaches, merging neighboring cliques does not decrease $d_a(\boldsymbol{q}_{\mathrm{ER}}^{(\mathrm{Mod})}, \boldsymbol{b}(T_{k,s}))$. This demonstrates that these two approaches effectively address the granularity problem for the ring of cliques. Let us note that both these approaches heavily rely on the hyperspherical geometry, and that they cannot be applied to the modularity function directly. Indeed, the first approach does not work because maximizing ERM$(T; A, \gamma)$ as a function of γ yields the trivial solution $\gamma = 0$, while the second approach is ill-defined without the notion of latitude.

While these two approaches effectively address the granularity problem of modularity for this ring of cliques network, they do not work well in general. We provide a better and more universal approach in Section 5.2.

3.4 Likelihood of generalized planted partition models

The Planted Partition Model (PPM) is one of the simplest random graph models that incorporates community structure. In this model, we assume there is some planted clustering into communities (the ground truth partition) and that nodes of the same community are connected with probability p_{in}, while nodes of different communities are connected with probability $p_{\mathrm{out}} < p_{\mathrm{in}}$. The likelihood of the PPM was derived in Holland et al. (1983). For an adjacency matrix A, the likelihood that it was generated by a PPM with clustering C, is given by

$$\mathrm{PPM} - \mathrm{Likelihood}(A|C, p_{\mathrm{in}}, p_{\mathrm{out}})$$
$$= \prod_{ij \in \mathrm{Intra}(C)} p_{\mathrm{in}}^{A_{ij}}(1 - p_{\mathrm{in}})^{1 - A_{ij}} \prod_{ij \in \mathrm{Inter}(C)} p_{\mathrm{out}}^{A_{ij}}(1 - p_{\mathrm{out}})^{1 - A_{ij}}.$$

In this standard PPM, we see that the adjacency matrix has binary entries. In Avrachenkov et al. (2020), the PPM is generalized to allow for *pairwise interactions* in any measurable set $\mathcal{I}$. That is, we require $A_{ij} \in \mathcal{I}$ for all $i < j$. For example, by taking $\mathcal{I} = \mathbb{R}$, we get a weighted undirected graph, while directed graphs can be modeled by $\mathcal{I} = \mathbb{R}^2$, so that each interaction corresponds to the tuple of weights w_{ij} and w_{ji}. Furthermore, this generalization also allows one to model temporal and multilayer networks. Similarly to the binary PPM, it is assumed that the distribution of the interaction between i and j only depends on whether i and j are in the same community. That is, there are likelihood functions $f_{\mathrm{in}}, f_{\mathrm{out}}$ that measure the likelihood of an interaction $A_{ij} \in \mathcal{I}$ resulting from an intra- or inter-community interaction. Importantly, we require the interactions to be pairwise independent. This allows us to express the likelihood of the interaction matrix A as the product of these pairwise likelihoods:

$$\mathrm{GenPPM} - \mathrm{Likelihood}(A|C, f_{\mathrm{in}}, f_{\mathrm{out}})$$
$$= \prod_{ij \in \mathrm{Intra}(C)} f_{\mathrm{in}}(A_{ij}) \prod_{ij \in \mathrm{Inter}(C)} f_{\mathrm{out}}(A_{ij}).$$

After taking the logarithm, it is easy to see that this is equal to the maximization variant of correlation clustering with $w_{ij}^+ = \log f_{\mathrm{in}}(A_{ij})$ and $w_{ij}^- = \log f_{\mathrm{out}}(A_{ij})$, so that, by Lemma 1, it is a projection method with query vector

$$q_{ij}^{(\text{PPM})} = \log \frac{f_{\text{in}}(A_{ij})}{f_{\text{out}}(A_{ij})}. \tag{15}$$

The standard binary PPM is recovered for $f_{\text{in}}(a) = p_{\text{in}}^a (1 - p_{\text{in}})^{1-a}$ and $f_{\text{out}}(a) = p_{\text{out}}^a (1 - p_{\text{out}})^{1-a}$.

3.4.1 The granularity bias of likelihood maximization

It has been observed that likelihood maximization methods for community detection have a bias towards communities of sizes close to log n (Zhang and Peixoto, 2020; Peixoto, 2021; Gösgens et al., 2023b). This can be understood by linking likelihood maximization to Bayesian inference (Peixoto, 2021): Bayesian community detection methods (Peixoto, 2019) assume a prior distribution over the set of clusterings and then find the clustering with the highest posterior probability. Bayes' rule reads

$$\text{Posterior}(C|A) = \frac{\text{Likelihood}(A|C) \cdot \text{Prior}(C)}{\sum_{C'} \text{Likelihood}(A|C') \cdot \text{Prior}(C')}.$$

Note that the denominator is constant w.r.t. C. If we were to assume a uniform prior, i.e., Prior(C) ∝ 1, then we get Posterior ($C|A$) ≡Likelihood ($A|C$). This tells us that likelihood maximization is equivalent to Bayesian inference under the assumption of a uniform prior. The uniform distribution over clusterings has been studied in the field of combinatorics for decades (Harper, 1967; Sachkov, 1997). For example, it is known that, asymptotically as $n \rightarrow \infty$, almost all clusters will have sizes close to log n. This explains why likelihood maximization methods have a bias towards clusterings of this granularity.

3.5 Which methods are *not* projection methods?

Not every community detection method fits our hyperspherical framework of community detection. A community detection method is *not* a projection method if the quality function cannot be monotonously transformed to a sum over intra-cluster pairs. Also, in projection methods, the contribution of a node-pair ij to the sum may depend on the input data (e.g., the graph), but *not* on the clustering C. In this section, we give a few examples of methods that do not fit this framework.

3.5.1 Other inferential methods

In Section 3.4 we saw that some likelihood methods fit our hyperspherical framework. However, not *all* inferential methods are projection methods. For example, suppose we take a PPM where the intra-community density is such that each node has λ intra-community neighbors in expectation. Then, if i and j are in a community of size s, they should be connected with probability $\lambda/(s - 1)$. Hence, the likelihood function f_{in} in (15) would depend on s, which requires the query vector $q^{(\text{PPM})}$ to depend on C. Our hyperspherical framework does not allow for this. Similarly, the Bayesian stochastic blockmodeling inference from Peixoto (2019) does not fit our hyperspherical framework because the contribution of each node pair ij depends on the size (and label) of the communities of i and j in an intricate way that cannot be captured by a query vector.

3.5.2 *k*-means clustering

The *k-means algorithm* is arguably the oldest and most well-studied clustering method (Jain, 2010). The aim of *k*-means is to divide given vectors $x_1, \ldots, x_n \in \mathbb{R}^d$ into k clusters. For each cluster, we compute the *center* as the arithmetic mean of the vectors inside this cluster. The *k*-means algorithm iteratively computes the centers and re-assigns each vector to its nearest center until convergence. In Dhillon et al. (2004) it is shown that *k*-means is equivalent to maximizing

$$\text{Trace}\left(\hat{H}(C)^{\top} X \hat{H}(C)\right), \tag{16}$$

where $X \in \mathbb{R}^{n \times n}$ is defined by $X_{ij} = \langle x_i, x_j \rangle$, and $\hat{H}(C) = H(C)(H(C)^{\top} H(C))^{-1/2} \in \mathbb{R}^{n \times k}$. That is, $\hat{H}(C)_{ia} = s_a^{-1/2}$ if node i is part of the community with label a and size s_a. Note that this form resembles the trace-maximization of Markov stability. A straightforward computation shows that

$$\left(\hat{H}(C)^{\top} X \hat{H}(C)\right)_{aa} = \frac{1}{s_a} \left(H(C)^{\top} X H(C)\right)_{aa}.$$

Therefore, the contribution of each community is again normalized by its size, which is not allowed in our hyperspherical framework.

Another clustering method closely related to *k*-means is *spectral clustering* (Von Luxburg, 2007). In spectral clustering, we are given an affinity matrix $X \in \mathbb{R}^{n \times n}$ and consider its leading eigenvectors. These leading eigenvectors define coordinates for the n objects, which are then clustered using spatial clustering methods like *k*-means. Spectral clustering differs from other clustering and community detection methods in the sense that it does not explicitly optimize a quality function. However, when using *k*-means for the final clustering step, one could consider it to be optimizing something of the form of (16), with X replaced by its low-rank approximation. Therefore, spectral clustering also does not fit our framework of projection methods.

4 Projection algorithms

The previous section shows that many community detection methods fit our definition of projection methods. A consequence of this is that the same optimization algorithms can be used for each of them. However, it is known that this optimization is NP-hard, and in some forms even APX-hard.

4.1 Exact optimization

The general problem of correlation clustering (Bansal et al., 2004) and the subproblem of maximizing modularity (Brandes et al., 2007; Meeks and Skerman, 2020) are known to be NP-complete. However, modularity maximization is known to be Fixed-Parameter Tractable (FPT) when parametrized by the size of the minimum vertex cover of the graph (Meeks and Skerman,

2020). Nevertheless, it has been shown that modularity maximization (Dinh et al., 2015) and the maximization variant of correlation clustering (Charikar et al., 2005) are APX-hard, meaning that approximating it to any constant factor is NP-hard. This tells us that for large graphs, it may be prohibitively expensive to compute the clustering that minimizes $d_a(\boldsymbol{q}, \boldsymbol{b}(C))$ over the set of clusterings C, for a general query vector $\boldsymbol{q}$.

Nevertheless, there are some approaches that are able to optimize some of the objectives from Section 3 with surprising efficiency. In particular, the *Bayan algorithm* (Aref et al., 2022) for modularity maximization is able to find the exact modularity maximum in graphs of up to several thousands nodes within hours. This approach relies on an Integer Linear Programming (ILP) formulation. Similar ILP formulations exist for the general problem of correlation clustering (Bansal et al., 2004). These can be converted to the following general ILP formulation of the projection method: in the projection step, we maximize

$$\sum_{i<j} \boldsymbol{q}_{ij} \boldsymbol{b}(C)_{ij},$$

subject to $\boldsymbol{b}(C)_{ij} \in \{-1, 1\}$ for all $i < j$ and the constraints

$$b_{ij} + b_{ik} - b_{jk} \leq 1,$$

for all $i, j, k \in [n]$.

4.2 Approximate optimization

There are many approximate maximization algorithms that are able to quickly find clusterings with high modularity. The Louvain (Blondel et al., 2008) and Leiden (Traag et al., 2019) algorithms are perhaps the most well-known heuristics for modularity maximization. These algorithms iterate over the nodes and make use of the network sparsity to find the greedy relabeling of a node. For a node i, finding this greedy relabeling has complexity $\mathcal{O}(d_i)$. The Louvain algorithm terminates when it achieves a local maximum. Since it is non-trivial to bound the number of iterations needed to reach a local maximum, there are no theoretical guarantees for the complexity. However, the running time empirically scales linearly with the number of edges. While these algorithms often attain values close to the global optimum, they rarely find the exact global optimum (Aref et al., 2023).

The algorithms proposed in the field of correlation clustering come with theoretical approximation guarantees. Due to the equivalence established in Lemma 1, these algorithms can be applied to modularity maximization. Conversely, modularity maximization algorithms like the Louvain algorithm can be applied to the correlation clustering quality function, allowing for comparisons between these algorithms. Interestingly, while there exist no optimization guarantees for the Louvain algorithm, it does seem to outperform correlation clustering algorithms in such comparisons (Veldt et al., 2018). The Louvain algorithm can be modified to minimize $d_a(\boldsymbol{q}, \boldsymbol{b}(C))$ with similar performance. However, the computational complexity may depend on the particular query vector $\boldsymbol{q}$ (Gösgens et al., 2023a). More precisely, our modification of Louvain assumes a query vector of the form $\boldsymbol{q} = \boldsymbol{v}(S + L)$, where $S \in \mathbb{R}^{n \times n}$ is a sparse matrix and $L \in \mathbb{R}^{n \times n}$ is a

low-rank matrix. This way, finding the greedy relabeling for a node i has linear complexity in terms of the number of non-zero entries of S adjacent to i. We observe that the running time of this re-implementation of Louvain is proportional to the number of elements in non-zero elements in S (Gösgens et al., 2023b).

We denote by $\mathcal{L}(\boldsymbol{q})$ the clustering vector that results from minimizing $d_a(\boldsymbol{q}, \boldsymbol{b}(C))$ over the set of clusterings C by the Louvain algorithm.

4.3 Do we *need* the global optimum?

The modularity landscape is known to be *glassy* (Good et al., 2010), which means that there are many local maxima with values close to the global maximum. It is likely that for a general query vector $\boldsymbol{q}$, the landscape of $d_a(\boldsymbol{q}, \boldsymbol{b}(C))$ suffers from a similar glassiness, which explains why its exact minimization is computationally expensive, while its approximate minimization is computationally cheap.

However, the ultimate goal of community detection is not to minimize the distance to some query vector, but to obtain a meaningful clustering of the network nodes. In settings where we have a generative model, like the PPM, the popular LFR benchmark (Lancichinetti et al., 2008), or the more recent ABCD benchmark (Kamiński et al., 2021), a meaningful clustering is a clustering that is similar to the planted clustering. However, there is no guarantee that the planted clustering corresponds to the global (or even a local) optimum. Most prominently, in sparse network models, it is highly unlikely that a locally optimal clustering corresponds to the planted clustering. A simple argument for this is that a sparse network model contains isolated nodes with high probability, and these nodes will not be assigned to their true community in any locally optimal clustering.

Moreover, when applying the Louvain algorithm to graphs from generators, it has been observed that the obtained modularity often exceeds the modularity of the planted clustering. This tells us that simple greedy optimization algorithms like Louvain already result in a clustering vector $\mathcal{L}(\boldsymbol{q})$ that is nearer to $\boldsymbol{q}$ than the planted clustering vector $\boldsymbol{b}(T)$, i.e., $d_a(\boldsymbol{q}, \mathcal{L}(\boldsymbol{q})) \leq d_a(\boldsymbol{q}, \boldsymbol{b}(T))$.

In the experiments for this paper, we have applied the Louvain algorithm 4,150 times for different combinations of query mappings and networks. In most of these cases, we chose a query vector using a heuristic, which will be explained in Section 5.2, that is designed to ensure $d_a(\boldsymbol{q}, \mathcal{L}(\boldsymbol{q})) \approx d_a(\boldsymbol{q}, \boldsymbol{b}(T))$. However, in all these 4,150 applications of the Louvain algorithm, we have only observed 253 cases where $d_a(\boldsymbol{q}, \mathcal{L}(\boldsymbol{q})) > d_a(\boldsymbol{q}, \boldsymbol{b}(T))$. Furthermore, most of these might be attributed to numerical errors because there are only 10 instances where $d_a(\boldsymbol{q}, \boldsymbol{b}(C))$ is more than 1% larger than $d_a(\boldsymbol{q}, \boldsymbol{b}(T))$ and no instances where it is more than 2% larger. This tells us that approximate optimization algorithms like Louvain easily find clusterings with higher quality [i.e., lower $d_a(\boldsymbol{q}, \boldsymbol{b}(C))$] than the planted clustering.

The important conclusion from these observations is that better optimization algorithms do not necessarily result in more meaningful clusterings. Instead, it seems more important to choose the query vector so minimizers of $d_a(\boldsymbol{q}, \boldsymbol{b}(C))$ are close to the ground truth clustering vector $\boldsymbol{b}(T)$.

5 Choosing the query vector: heuristics and open problems

In the previous section, we have seen that the choice of the quality function (or equivalently, the query vector) may have a bigger impact on the performance of a community detection method than the choice of the optimization algorithm.

Choosing a quality function is difficult because it is hard to compare two quality functions in a meaningful way. However, when restricting to the set of projection methods, the hyperspheric geometry provides us with additional tools to compare the query vectors: for example, we can compute the latitude of the query vector and distances between query vectors. This provides us with some information about the relative position of the query vectors. In addition, these query vectors define a vector space, so that linear combinations of query vectors also correspond to community detection methods. In this section, we discuss several ways to choose a query mapping, which maps graphs to a query vectors q.

5.1 Graph generators

We assume that we are given some generator, which produces a tuple (A, T) of an adjacency matrix and a planted clustering (the ground truth). This generator defines a joint distribution on (A, T). In our experiments, we make use of several different generators:

5.1.1 The planted partition model (PPM)

The standard (not generalized) PPM from Section 3.4 is the simplest random graph model with community structure. In this model, there is a planted clustering (partition) of the nodes, and nodes of the same community are more likely to connect to each other than nodes of different communities. We discuss three different variants of the PPM. The first one is a random graph model with homogeneity in both the degree and the community size distribution. We consider k equally sized communities of size n/k (assuming k divides n), and assume that each node has (in expectation) the same number of neighbors inside and outside its community, given by λ_{in} and λ_{out}, respectively. We then set the connection probabilities as

$$p_{\text{in}} = \frac{\lambda_{\text{in}}}{s - 1}, \quad \text{and} \quad p_{\text{out}} = \frac{\lambda_{\text{out}}}{n - s}.$$

This way, each node's degree follows the same distribution, which is a sum of two binomially distributed random variables, which can be approximated by a Poisson distribution with mean $\lambda_{\text{in}} + \lambda_{\text{out}}$.

5.1.2 The heterogeneously-sized PPM (HPPM)

The second variant of the PPM has homogeneous degrees (again, approximately Poisson distributed), but has heterogeneity in the community-size distribution. We draw k community sizes from a power-law distribution with some power-law exponent δ, meaning that the probability of obtaining a size s decays as $s^{-\delta}$. We make sure that each node has on average λ_{in} intra-community neighbors and λ_{out} neighbors outside of its community, by setting

$$p_{\text{in}}(s) = \frac{\lambda_{\text{in}}}{s - 1}$$

for nodes in communities of size s, and

$$p_{\text{out}} = \frac{n\lambda_{\text{out}}}{2 \cdot (N - m_T)},$$

where m_T is the number of intra-community pairs in the planted clustering T.

5.1.3 The degree-corrected PPM (DCPPM)

To obtain a graph generator with degree heterogeneity and homogeneous community sizes, we assign a weight $\theta_i > 0$ to each node and use the PPM parametrization that was proposed in Prokhorenkova and Tikhonov (2019). We consider k equally-sized communities of size n/k (again, assuming k divides n). We denote the sum of weights inside the a-th community by Θ_a and denote the total weight by $\Theta = \sum_{a=1}^{k} \Theta_a$. Nodes i and j that are both in the a-th community are connected with probability

$$p_{\text{in}}(\theta_i, \theta_j, \Theta_a) = \frac{\lambda_{\text{in}}}{\lambda_{\text{in}} + \lambda_{\text{out}}} \frac{\theta_i \theta_j}{\Theta_a} + \frac{\lambda_{\text{out}}}{\lambda_{\text{in}} + \lambda_{\text{out}}} \frac{\theta_i \theta_j}{\Theta},$$

and nodes from different communities are connected with probability

$$p_{\text{out}}(\theta_i, \theta_j) = \frac{\lambda_{\text{out}}}{\lambda_{\text{in}} + \lambda_{\text{out}}} \frac{\theta_i \theta_j}{\Theta}.$$

With these parameters, a node has on average approximately λ_{in} neighbors inside its community and λ_{out} neighbors outside its community. In addition, the expected degree of a node i is approximately equal to its weight θ_i. To obtain a degree distribution with power-law exponent τ, we draw the weights from a distribution with this same power-law exponent.

5.1.4 The artificial benchmark for community detection (ABCD)

The Artificial Benchmark for Community Detection (ABCD) is a graph generator that incorporates heterogeneity in both the degree and community-size distribution in order to generate graphs that resemble real-world networks (Kamiński et al., 2021). This is done by generating a sequence of community sizes and degrees with power-law exponents δ and τ. Then, it performs a matching process to assign degrees to nodes inside communities. The generator has a parameter ξ that controls the fraction of edges that are inter-community edges.

5.1.5 Parameter choices in our graph generators

We set the parameters of the graph generators as follows: we consider graphs with $n = 1,000$ nodes and mean degree $\lambda_{\text{in}} + \lambda_{\text{out}} = 8$. We choose the parameters of these generators so that each node has (in expectation) $\lambda_{\text{out}} = 2$ neighbors outside its community. For DCPPM and ABCD, we set the power-law exponent of the degree distribution to $\tau = 2.5$. We generate the planted partitions as follows: For PPM and DCPPM, we consider $k = 50$ communities of size $s = 20$ each. For ABCD, we set $\xi = \frac{1}{4}$.

Granularity errors of Markov stability on PPM networks.

Performance of Markov stability on PPM networks.

FIGURE 3
For 50 graphs drawn from the Planted Partition Model (PPM), we evaluate Markov stability with time $t \in [5]$, and compare the clusterings that are obtained with and without applying the granularity heuristic. $q^*(\cdot)$ denotes the heuristic. A positive granularity error indicates that the detected clustering is coarser than the planted clustering. $\rho(C, T) = \cos d_\rho(\boldsymbol{b}(C), \boldsymbol{b}(T))$ is the Pearson correlation between the clustering vectors, which we wish to make close to 1. We see that $\boldsymbol{q}^*(\boldsymbol{q}_t^{(MS)})$ strongly outperforms $\boldsymbol{q}_t^{(MS)}$. **(A)** Granularity errors of Markov stability on PPM networks. **(B)** Performance of Markov stability on PPM networks.

5.2 A heuristic for controlling the granularity of the detected clustering

Modularity and Markov stability both have a parameter that controls the granularity of the detected clusterings. Modularity comes with a resolution parameter γ, and increasing γ typically results in detecting communities of smaller sizes. However, it is unclear how this resolution parameter should be chosen in order to detect clusterings of the desired granularity. With 'desired', we mean that the granularity of the detected clustering is similar to the granularity of the planted clustering in cases where the graph is drawn from a graph generator. For ER-modularity, there is a particular value of $\gamma(p_{\text{in}}, p_{\text{out}})$ for which maximizing ER-modularity is equivalent to maximizing the likelihood of a PPM with parameters $p_{\text{in}}, p_{\text{out}}$. However, as mentioned in Section 3.4, it is known that maximizing this likelihood is biased towards communities of logarithmic size. Markov stability comes with a time parameter t which controls the granularity of the detected clustering. Increasing t results in detecting larger communities. Again, it is unclear how this time should be chosen in order to detect communities of the desired granularity.

Within the framework of projection methods, a natural measure of the granularity of a clustering C is the latitude $\ell(\boldsymbol{b}(C))$ of the corresponding clustering vector. Hence, in cases where the graph is drawn from a generator with a planted clustering T, a clustering with
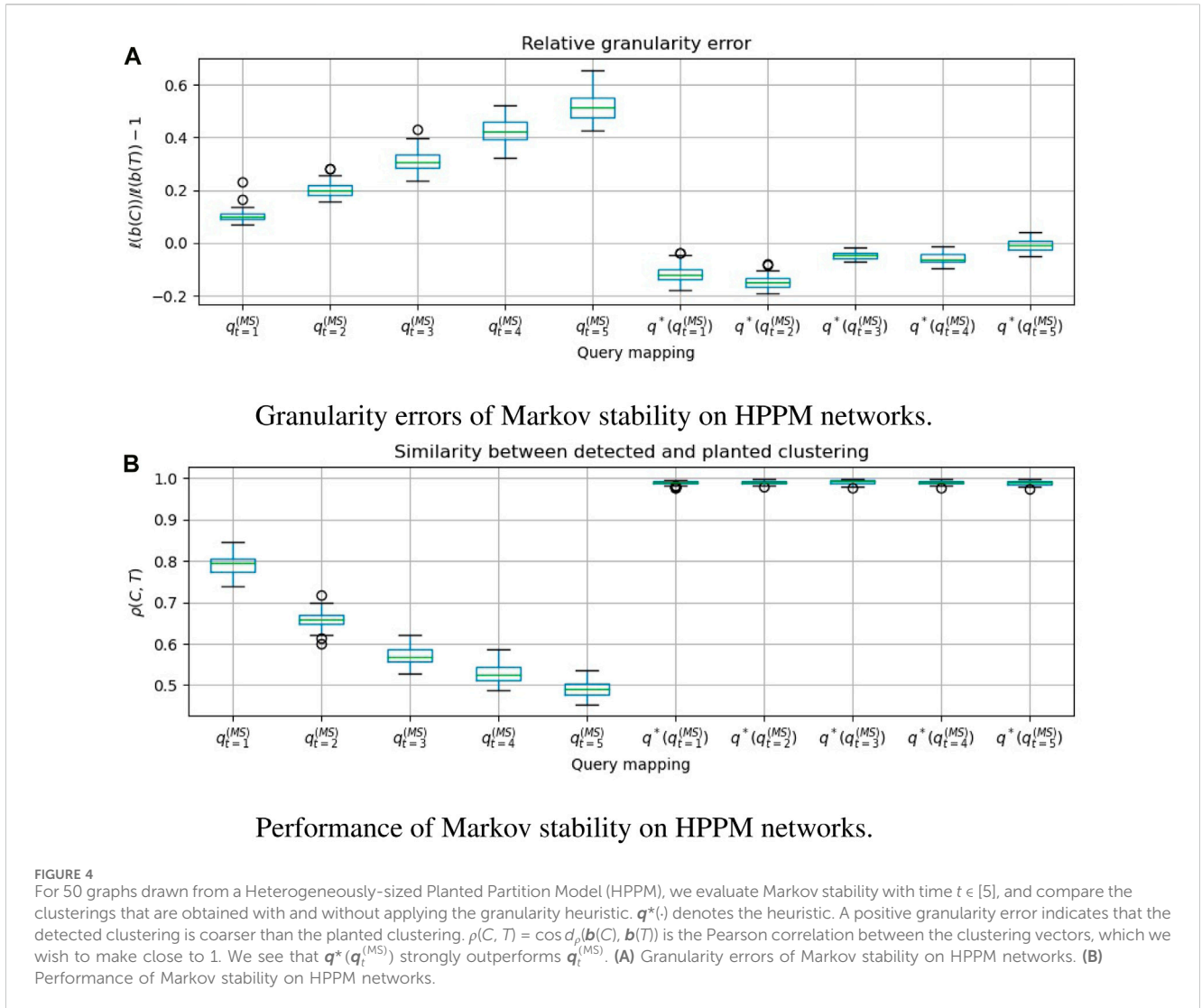
'desired' granularity is a clustering C with $\ell(\boldsymbol{b}(C)) \approx \ell(\boldsymbol{b}(T))$. In turn, the desired $\ell(\boldsymbol{b}(C))$ can be obtained by choosing the right latitude of a query vector. How to make this choice is the topic of the remainder of this Section 5.2.

The simplest way to change a query vector in order to detect clusterings of coarser granularity, is to add a multiple of $\boldsymbol{1}$. That is, a new query vector $\boldsymbol{q}' = \boldsymbol{q} + c \cdot \boldsymbol{1}$ for some $c > 0$. The vectors $\boldsymbol{q}'$ and $\boldsymbol{q}$ lie on the same meridian, i.e., $d_\rho(\boldsymbol{q}, \boldsymbol{q}') = 0$, while $\boldsymbol{q}'$ is further away from $-\boldsymbol{1}$, so that $\ell(\boldsymbol{q}') > \ell(\boldsymbol{q})$. Hence, adding $c \cdot \boldsymbol{1}$ is equivalent to projecting the vector $\boldsymbol{q}$ to a different latitude, i.e., $\boldsymbol{q}' = \mathcal{P}_\lambda(\boldsymbol{q})$ for some $\lambda \in (\ell(\boldsymbol{q}), \pi]$. Similarly, the simplest way to change a query vector in order to detect clusterings of finer granularity, is to subtract a multiple of $\boldsymbol{1}$, i.e., $\boldsymbol{q}' = \boldsymbol{q} - c \cdot \boldsymbol{1}$, which is equivalent to $\boldsymbol{q}' = \mathcal{P}_\lambda(\boldsymbol{q})$ for $\lambda \in [0, \ell(\boldsymbol{q}))$. Hence, the question becomes: given a query vector $\boldsymbol{q}$, how should λ be chosen such that $\boldsymbol{q}' = \mathcal{P}_\lambda(\boldsymbol{q})$ results in a clustering with similar granularity as $\boldsymbol{b}(T)$, i.e., such that $\ell(\mathcal{L}(\boldsymbol{q}')) \approx \ell(\boldsymbol{b}(T))$?

In Gösgens et al. (2023a), we proposed a general heuristic that prescribes this latitude as a function of $\lambda_T = \ell(\boldsymbol{b}(T))$ and $\theta = d_\rho(\boldsymbol{q}, \boldsymbol{b}(T))$. This *granularity heuristic* prescribes

$$\lambda^\star(\lambda_T, \theta) = \arccos\left(\frac{\cos \lambda_T \cos \theta}{1 + \sin \lambda_T \sin \theta}\right) \qquad (17)$$

We denote the query vector obtained by applying the granularity heuristic, by $\boldsymbol{q}^*(\boldsymbol{q}) = \mathcal{P}_{\lambda^\star(\lambda_T, \theta)}(\boldsymbol{q})$, where $\boldsymbol{q}$ can be any query vector.

FIGURE 4
For 50 graphs drawn from a Heterogeneously-sized Planted Partition Model (HPPM), we evaluate Markov stability with time $t \in [5]$, and compare the clusterings that are obtained with and without applying the granularity heuristic. $\boldsymbol{q}^{\star}(\cdot)$ denotes the heuristic. A positive granularity error indicates that the detected clustering is coarser than the planted clustering. $\rho(C, T) = \cos d_\rho(\boldsymbol{b}(C), \boldsymbol{b}(T))$ is the Pearson correlation between the clustering vectors, which we wish to make close to 1. We see that $\boldsymbol{q}^{\star}(\boldsymbol{q}_t^{(MS)})$ strongly outperforms $\boldsymbol{q}_t^{(MS)}$. **(A)** Granularity errors of Markov stability on HPPM networks. **(B)** Performance of Markov stability on HPPM networks.

The latitude in (17) is chosen so that $d_a(\boldsymbol{q}^{\star}, \boldsymbol{b}(T)) = \theta$. We have observed that this leads to $d_a(\boldsymbol{q}^{\star}, \mathcal{L}(\boldsymbol{q}^{\star})) \approx \theta$ and $d_\rho(\boldsymbol{q}^{\star}, \mathcal{L}(\boldsymbol{q}^{\star})) \approx \theta$. Whenever these approximations are valid, it can be shown that $\ell(\mathcal{L}(\boldsymbol{q}^{\star})) \approx \ell(\boldsymbol{b}(T))$.

We briefly illustrate how solving $d_a(\boldsymbol{q}^{\star}, \boldsymbol{b}(T)) = \theta$ leads to (17): we use (4) to express $\cos \theta$ in terms of λ_T, $\lambda^{\star}$ and $d_a(\boldsymbol{q}^{\star}, \boldsymbol{b}(T))$ like

$$\cos \theta = \frac{\cos d_a(\boldsymbol{q}^{\star}, \boldsymbol{b}(T)) - \cos \lambda_T \cdot \cos \lambda^{\star}}{\sin \lambda_T \cdot \sin \lambda^{\star}}.$$

Squaring both sides and making the substitutions $\cos d_a(\boldsymbol{q}^{\star}, \boldsymbol{b}(T)) = \cos \theta$ and $\sin^2 \lambda^{\star} = 1 - \cos^2 \lambda^{\star}$ yields

$$\cos^2 \theta = \frac{\cos^2 \theta + \cos^2 \lambda_T \cdot \cos^2 \lambda^{\star} - 2 \cos \theta \cdot \cos \lambda_T \cdot \cos \lambda^{\star}}{\sin^2 \lambda_T \cdot (1 - \cos^2 \lambda^{\star})}.$$

This can be rewritten to the following quadratic equation in $\cos \lambda^{\star}$:

$$(1 - \sin^2 \theta \sin^2 \lambda_T)\cos^2 \lambda^{\star} - 2 \cos \theta \cdot \cos \lambda_T \cdot \cos \lambda^{\star} + \cos^2 \theta \cdot \cos^2 \lambda_T = 0,$$

which has solutions

$$\cos \lambda^{\star} = \frac{2 \cos \theta \cdot \cos \lambda_T \pm \sqrt{4 \cos^2 \theta \cdot \cos^2 \lambda_T - 4 \cdot (1 - \sin^2 \theta \sin^2 \lambda_T)\cos^2 \theta \cdot \cos^2 \lambda_T}}{2 \cdot (1 - \sin^2 \theta \sin^2 \lambda_T)}$$

$$= \cos \theta \cdot \cos \lambda_T \cdot \frac{1 \pm \sin \theta \sin \lambda_T}{1 - \sin^2 \theta \sin^2 \lambda_T}$$

$$= \frac{\cos \theta \cdot \cos \lambda_T}{1 \mp \sin \theta \sin \lambda_T}.$$

This gives two possible solutions for $\lambda^{\star}$, one of which corresponds to (17). We refer to Gösgens et al. (2023b) for the remaining details of the derivation and the experimental validation of this heuristic.

Note that $\cos \theta$ is the Pearson correlation coefficient between $\boldsymbol{q}$ and $\boldsymbol{b}(T)$, and can be considered a measure of how much information $\boldsymbol{q}$ carries of the clustering T. As a special case, note that $\cos \theta = 1$ implies that $\boldsymbol{q}$ and $\boldsymbol{b}(T)$ lie on the same meridian, and we can see that $\lambda^{\star}(\lambda_T, 0) = \lambda_T$, so that $\boldsymbol{q}^{\star} = \boldsymbol{b}(T)$. In the other extreme, where $\boldsymbol{q}$ is not correlated with $\boldsymbol{b}(T)$ (i.e., $\theta = \pi/2$), we have $\lambda^{\star}(\lambda_T, \pi/2) = \pi/2$, so that the resulting query vector lies on the equator (just like the modularity vector for $\gamma = 1$). For $\theta \in (0, \pi/2)$, the heuristic latitude $\lambda^{\star}(\lambda_T, \theta)$ is between λ_T and $\pi/2$.

To compute the heuristic latitude choice in (17), we need estimates of λ_T and θ, which requires some knowledge of the
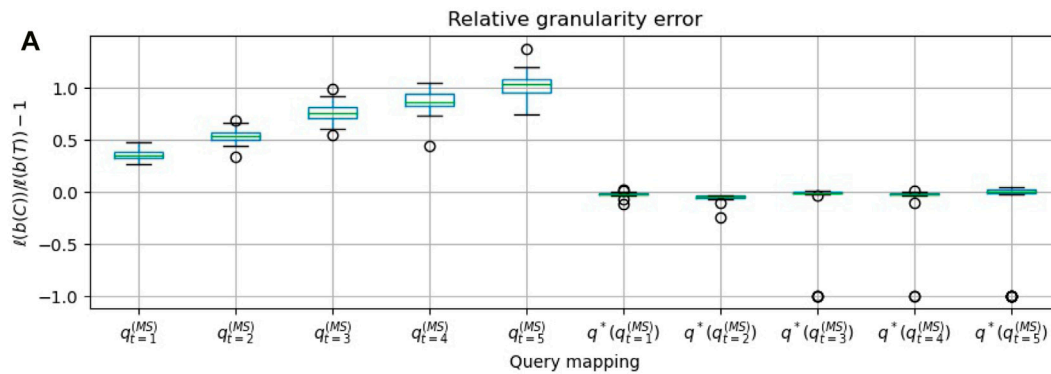
Granularity errors of Markov stability on DCPPM networks.
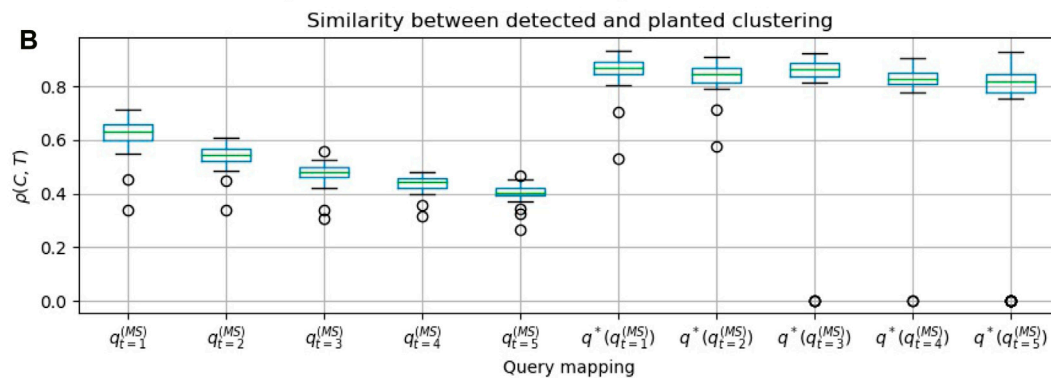
Performance of Markov stability on DCPPM networks.

FIGURE 5
For 50 graphs drawn from the Degree-Corrected Planted Partition Model (DCPPM), we evaluate Markov stability with time $t \in [5]$, and compare the clusterings that are obtained with and without applying the granularity heuristic. $\boldsymbol{q}^*(\cdot)$ denotes the heuristic. A positive granularity error indicates that the detected clustering is coarser than the planted clustering. $\rho(C, T) = \cos d_\rho(\boldsymbol{b}(C), \boldsymbol{b}(T))$ is the Pearson correlation between the clustering vectors, which we wish to make close to 1. We see that $\boldsymbol{q}^*(\boldsymbol{q}_t^{(MS)})$ strongly outperforms $\boldsymbol{q}_t^{(MS)}$. **(A)** Granularity errors of Markov stability on DCPPM networks. **(B)** Performance of Markov stability on DCPPM networks.

planted clustering T. It might seem that requiring this knowledge of the planted clustering defeats the purpose of community detection. However, requiring partial knowledge of the planted clustering is not uncommon in other community detection methods, such as likelihood-based methods (Newman, 2016; Prokhorenkova and Tikhonov, 2019). Moreover, when we have access to the graph generator, we can use this to estimate the means of λ_T and θ, and use these estimates in (17).

In Gösgens et al. (2023a), we have shown that this granularity heuristic works well for several query vectors $\boldsymbol{q}$, including modularity vectors. In this section, we demonstrate that this heuristic also works well for Markov stability vectors. For $t \in \{1, \ldots, 5\}$, we consider the projection method with query mapping $\boldsymbol{q}_t^{(MS)}$ from (11), which is equivalent to maximizing the Markov stability for a discrete-time random walk with time t. We compare this method to the projection method with query mapping $\boldsymbol{q}^*(\boldsymbol{q}_t^{(MS)})$, which corresponds to applying our granularity heuristic to the Markov stability vector.

To quantify the quality of the approximation $\ell(\boldsymbol{b}(C)) \approx \ell(\boldsymbol{b}(T))$, we define the *relative granularity error* as $\ell(\boldsymbol{b}(C))/\ell(\boldsymbol{b}(T)) - 1$, which we want to be close to 0. Positive values indicate that the detected clustering is more coarse-grained than the planted clustering, while negative values indicate that the detected clustering is too fine-grained. We measure the similarity between the detected and planted communities by the correlation coefficient $\rho(C, T) = \cos d_\rho(\boldsymbol{b}(C), \boldsymbol{b}(T))$. Values close to 1 indicate that the clusterings are highly similar, while values close to zero indicate that C is not more similar to T than a random relabeling of C.

For the PPM, HPPM, DCPPM and ABCD graph generators, the results are shown in Figures 3–6. For each generator, we generate 50 graphs and show boxplots of the outcomes for each of the query mappings.

5.2.1 Effect of the heuristic

In Figures 3a, 4a, 5a, 6a, we see that the granularity heuristic indeed leads to detecting clusterings with granularity closer to the granularity of the planted clustering. For almost all cases, we see that
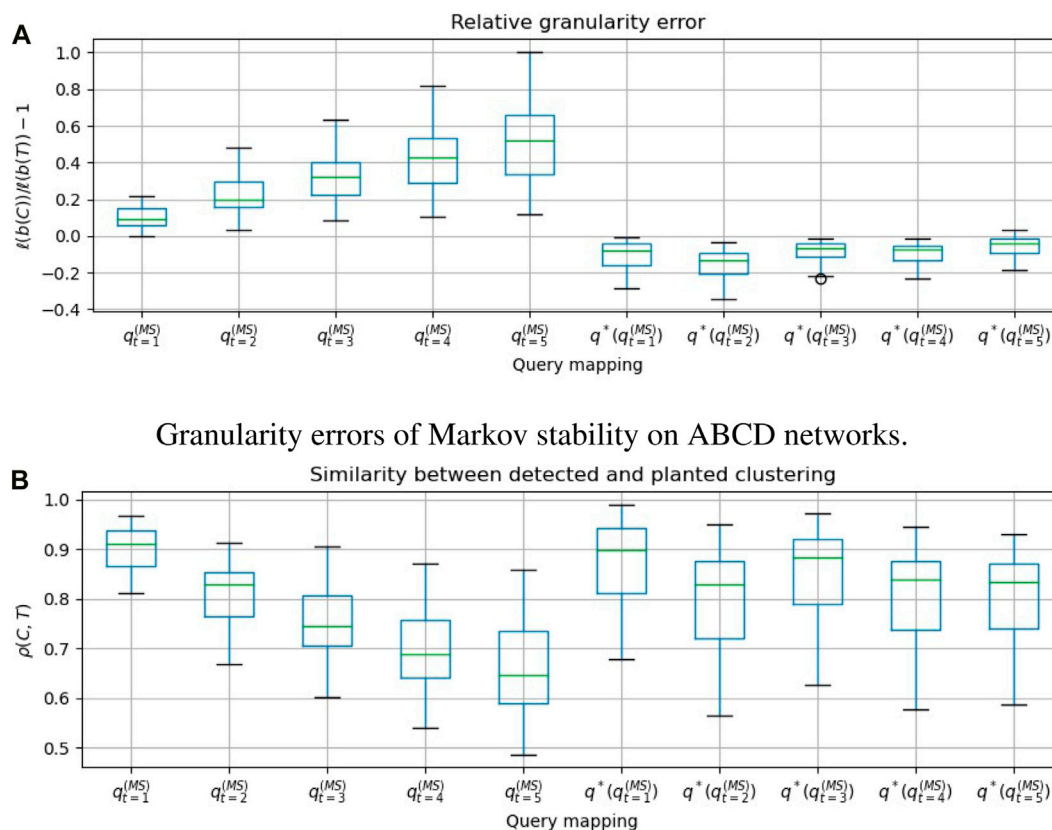
FIGURE 6
For 50 graphs drawn from the Artificial Benchmark for Community Detection (ABCD), we evaluate Markov stability with time $t \in [5]$, and compare the clusterings that are obtained with and without applying the granularity heuristic. $q^*(\cdot)$ denotes the heuristic. A positive granularity error indicates that the detected clustering is coarser than the planted clustering. $\rho(C, T) = \cos d_\rho(\boldsymbol{b}(C), \boldsymbol{b}(T))$ is the Pearson correlation between the clustering vectors, which we wish to make close to 1. We see that $q^*(\boldsymbol{q}_t^{(\mathrm{MS})})$ outperforms $\boldsymbol{q}_t^{(\mathrm{MS})}$ for $t \neq 1$. **(A)** Granularity errors of Markov stability on ABCD networks. **(B)** Performance of Markov stability on ABCD networks.

the median relative granularity error after applying the heuristic is closer to zero than before applying the heuristic. The only exception is $t = 1$ for the HPPM generator. We see that overall, the granularity heuristic results in clusterings that are slightly more fine-grained than the planted clustering.

In addition, Figures 3b, 4b, 5b, 6b show that the granularity heuristic typically leads to an increased similarity to the planted clustering. The only two exceptions are HPPM and ABCD for $t = 1$, where the granularity heuristic results in slightly lower performance. For the PPM generator, Figure 3B shows that for each of the values of t, the detection is near-perfect (all similarities are higher than $\rho = 0.97$). For the HPPM and DCPPM generators, Figures 4B, 5B show that the heterogeneity in the community sizes and degrees result in slightly lower performance on these generators.

5.2.2 Markov stability time and granularity

Figures 3a, 4a, 5a, 6a show that for the query vector $\boldsymbol{q}_t^{(\mathrm{MS})}$ (without applying the heuristic), larger values of t indeed lead to detecting more coarse-grained clusterings. However, $t = 1$ already results in clusterings that are more coarse-grained

than the planted clustering. We see similar outcomes in Figures 4a, 5a and (to a lesser extent) Figure 6A. Since we are using a discrete-time Markov chain, we cannot consider times $t \in (0, 1)$. For a continuous-time Markov chain, this is possible.

5.2.3 Sparsity and computation time

Note that the running time of the Louvain algorithm for Markov Stability vectors depends on the sparsity of the transition matrix $P(t)$. To illustrate: for $t = 1$, the transition matrix has the same number of positive entries as the adjacency matrix, and (our implementation of) the Louvain algorithm runs in around 5 s. In contrast, for $t = 5$, we have $P(5)_{ij} > 0$ for each pair of nodes that is connected by a path of length 5, which leads to a much denser matrix, and the Louvain algorithm takes roughly 200 s per instance. While the continuous-time variant of Markov stability may be able to detect clusterings of finer granularity, the corresponding transition matrix satisfies $P(t)_{ij} > 0$ whenever i and j are connected by *any* path. This leads to a much denser matrix, so that the Louvain algorithm will take even more time.
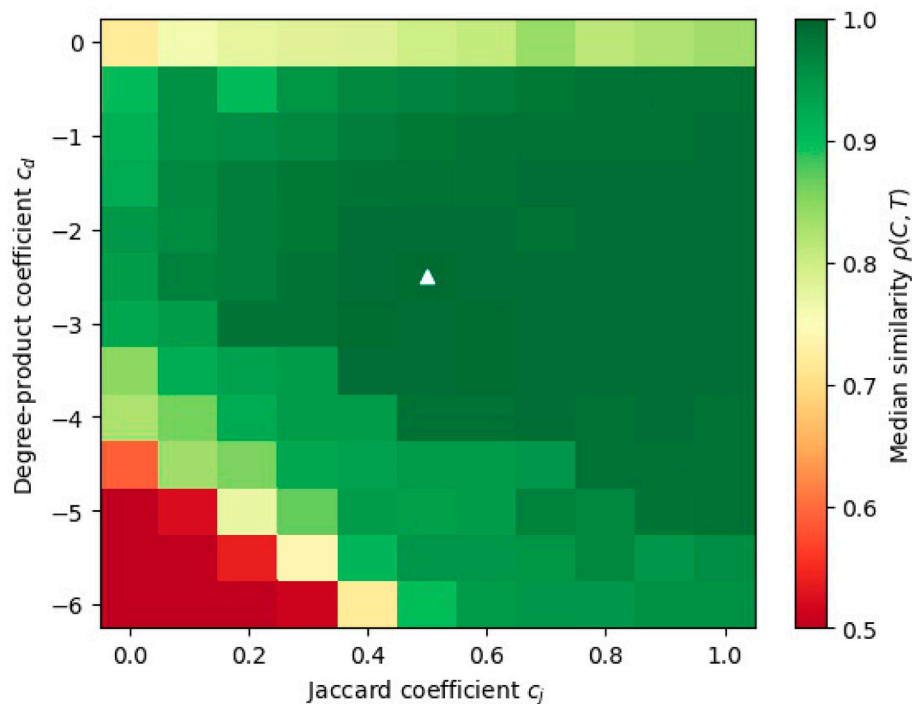
FIGURE 7
A heatmap of the median performance (similarity between the detected and planted clusterings, as measured by $\rho(C, T)$) for different linear combinations of query mappings. The medians are computed over 10 samples of ABCD graphs, with the same parameters as in the experiments of Section 5.2. The best performance is marked with a white triangle.

TABLE 1 The performance of several projection methods on a set of 50 ABCD graphs (the same set of graphs as in Figure 6).

	Markov stability					Likelihood		Grid search
	$t = 1$	$t = 2$	$t = 3$	$t = 4$	$t = 5$	PPM	DCPPM	
Median $\rho(C, T)$	0.912	0.830	0.746	0.689	0.646	0.809	0.801	0.973
Median $\ell(\boldsymbol{b}(C))/\ell(\boldsymbol{b}(T)) - 1$	0.094	0.196	0.325	0.428	0.522	−0.132	−0.180	−0.022

Note that Markov stability with $t = 1$ is equivalent to CL-modularity. The likelihood-based methods correspond to ER- and CL-modularity with the resolution parameter value for which they are equivalent to maximizing the PPM- and DCPPM-likelihood respectively (Newman, 2016). *Grid search* corresponds to the projection method obtained in Section 5.3.

5.3 Tuning a projection method to a graph generator

In Section 3, we have seen that the class of projection methods contains many interesting community detection methods. A further advantage of projection methods is that we can combine different community detection methods by taking linear combinations of their query mappings. On the one hand, this yields infinitely many community detection methods, and gives a lot of flexibility. On the other hand, this begs the question how to choose a suitable query mapping for a task at hand. For instance, are there preferable choices for a particular graph generator? In this section, we will partially address this question by showing how one can *tune* a projection method to a graph generator, in order to maximize the performance on graphs sampled from this generator.

We assume that we are given a generator that produces graphs with their community assignments. We consider linear combinations of a few query mappings and perform a *grid search* to find the coefficients that yield the best query vector. A grid search is a standard approach for hyperparameter tuning, where we discretize each of the parameter intervals and evaluate all possible combinations. To avoid overfitting, one usually generates two different sets of graphs: a *training* set and a *validation* set. The training set is used to find the best hyperparameter combination, while the validation set is used to get an unbiased estimate of the performance of the obtained hyperparameters values.

To demonstrate this method, we show how we can optimize a projection method for the ABCD graph generator from the previous section. To allow for comparison with this previous section, we consider the same 50 ABCD graphs as the validation set. For the training set, we generate 15 new ABCD graphs from this same generator.

We consider query vectors that are linear combinations of four vectors: the constant vector $\boldsymbol{1}$ (to control granularity), the adjacency vector $\boldsymbol{v}(A)$, the degree-product vector $\boldsymbol{d}(A)$ and the *Jaccard* vector $\boldsymbol{j}(A)$. The latter is defined as

follows: let $N(i)$ denote the neighborhood of i. We follow the convention that $i \in N(i)$ for all $i \in [n]$. Then $\boldsymbol{j}(A)_{ij}$ is the Jaccard similarity between the neighborhoods of i and j:

$$\boldsymbol{j}(A)_{ij} = \frac{|N(i) \cap N(j)|}{|N(i) \cup N(j)|}.$$

We consider query vectors that are linear combinations of these four vectors, $\boldsymbol{q} = c_1 \cdot \boldsymbol{1} + c_A \cdot \boldsymbol{v}(A) + c_d \cdot \boldsymbol{d}(A) + c_j \cdot \boldsymbol{j}(A)$. Since in the hyperspherical geometry, the length of the query vector does not affect the detected clustering, we can reduce the number of hyperparameters by one. Assuming that the best combination has $c_A > 0$, we set $c_A = 1$, thereby making the grid search more efficient. Moreover, for most values of the coefficient c_1, the detection method will result in clusterings of a wrong granularity. Thus, instead of fitting c_1, we use the granularity heuristic from Section 5.2. Then we are left with the following parametrization of query vectors:

$$\boldsymbol{q}(A; c_j, c_d) = \boldsymbol{q}^\star \big(\boldsymbol{v}(A) + c_j \cdot \boldsymbol{j}(A) + c_d \cdot \boldsymbol{d}(A) \big).$$

This leaves two hyperparameters to be tuned by the grid search. Note that the vector $\boldsymbol{d}(A)$ is a correction term for the degrees. Because of that, the best performance is likely to be found for $c_d \leq 0$. We choose the interval $c_d \in [-6, 0]$, which we discretize in steps of $\frac{1}{2}$. For the parameter c_j, we discretize the interval $[0,1]$ into steps of size $\frac{1}{10}$.

The results are shown in Figure 7. We see that there is a large region where the method performs well. In particular, the best-performing coefficients are $c_d = -\frac{5}{2}$ and $c_j = \frac{1}{2}$ with a median performance of $\rho = 0.993$ on the training set[2]. We apply this query mapping to the validation set and a median performance of $\rho = 0.973$, which is slightly lower than the performance on the training set, as expected due to selection bias.

Table 1 compares the performance of the trained projection method to other projection methods. We see that Markov stability with time $t = 1$ performs best among Markov stability methods (without the granularity heuristic), and also outperforms PPM- and DCPPM-likelihood maximization (Newman, 2016). Note that Markov stability with $t = 1$ is equivalent to CL-modularity maximization with $\gamma = 1$. This query mapping achieved a median performance of $\rho = 0.912$, which is good, but significantly lower than the performance of our optimized query mapping. In addition, we see that the optimized projection method outperforms the other projection methods in terms of latitude error due to the latitude heuristic. This is despite the fact that the likelihood-based methods require comparable knowledge about the generative process as our latitude heuristic.

In this demonstration, we have kept the setup relatively simple by taking combinations of only 4 vectors, reducing this to two coefficients. We did this for simplicity and so that we can visualize the performance on the training set by a two-dimensional

---

2 There are four combinations of coefficients that achieve this exact same median performance. We choose this one as it has the highest mean performance.

heatmap. This already led to strong performance. It is likely that we can improve performance even further by including a larger number of query vectors, and using a grid search instead of the granularity heuristics to determine the coefficient c_1. The obvious downside is that optimization for a larger number of parameters becomes computationally more demanding, and one must increase the size of the training set to avoid overfitting.

To conclude, we have shown that the class of projection methods unifies many popular community detection methods and is expressive enough to fit realistic benchmark generators like ABCD. This work paves the way to many follow-up research. On the one hand, there are many algorithmic questions: what projection algorithms work well for what query vectors? Are there sets of query vectors for which the minimization of d_a ($\boldsymbol{q}$, $\boldsymbol{b}(C)$) is *not* NP-hard? On the other hand, there are also several methodological questions: how do the best-performing coefficients of the linear combination depend on the network properties? For example, how does the best-performing coefficient c_d depend on the mean and variance of the degree distribution?

Data availability statement

Author contributions

Funding

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

The author(s) declared that they were an editorial board member of Frontiers, at the time of submission. This had no impact on the peer review process and the final decision.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

References

Aref, S., Chheda, H., and Mostajabdaveh, M. (2022). *The bayan algorithm: detecting communities in networks through exact and approximate optimization of modularity.* arXiv preprint arXiv:2209.04562.

Aref, S., Mostajabdaveh, M., and Chheda, H. (2023). *Heuristic modularity maximization algorithms for community detection rarely return an optimal partition or anything similar*. arXiv preprint arXiv:2302.14698.

Avrachenkov, K., Dreveton, M., and Leskelä, L. (2020). *Community recovery in non-binary and temporal stochastic block models. arXiv preprint arXiv:2008.04790.*

Bansal, N., Blum, A., and Chawla, S. (2004). Correlation clustering. *Mach. Learn.* 56, 89–113. doi:10.1023/b:mach.0000033116.57574.95

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of communities in large networks. *J. Stat. Mech. theory Exp.* 2008, P10008. doi:10.1088/1742-5468/2008/10/p10008

Brandes, U., Delling, D., Gaertler, M., Gorke, R., Hoefer, M., Nikoloski, Z., et al. (2008). On modularity clustering. *IEEE Trans. Knowl. data Eng.* 20, 172–188. doi:10.1109/tkde.2007.190689

Charikar, M., Guruswami, V., and Wirth, A. (2005). Clustering with qualitative information. *J. Comput. Syst. Sci.* 71, 360–383. doi:10.1016/j.jcss.2004.10.012

Chawla, S., Makarychev, K., Schramm, T., and Yaroslavtsev, G. (2015). "Near optimal lp rounding algorithm for correlationclustering on complete and complete k-partite graphs," in Proceedings of the forty-seventh annual ACM symposium on Theory of computing, 219–228.

Chung, F., and Lu, L. (2001). The diameter of sparse random graphs. *Adv. Appl. Math.* 26, 257–279. doi:10.1006/aama.2001.0720

Delvenne, J.-C., Yaliraki, S. N., and Barahona, M. (2010). Stability of graph communities across time scales. *Proc. Natl. Acad. Sci.* 107, 12755–12760. doi:10.1073/pnas.0903215107

Dhillon, I. S., Guan, Y., and Kulis, B. (2004). "Kernel k-means: spectral clustering and normalized cuts," in Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, 551–556.

Dinh, T. N., Li, X., and Thai, M. T. (2015). "Network clustering via maximizing modularity: approximation algorithms and theoretical limits," in 2015 IEEE International Conference on Data Mining (IEEE), 101–110.

Fortunato, S. (2010). Community detection in graphs. *Phys. Rep.* 486, 75–174. doi:10.1016/j.physrep.2009.11.002

Fortunato, S., and Barthélemy, M. (2007). Resolution limit in community detection. *Proc. Natl. Acad. Sci.* 104, 36–41. doi:10.1073/pnas.0605965104

Fortunato, S., and Hric, D. (2016). Community detection in networks: a user guide. *Phys. Rep.* 659, 1–44. doi:10.1016/j.physrep.2016.09.002

Good, B. H., De Montjoye, Y.-A., and Clauset, A. (2010). Performance of modularity maximization in practical contexts. *Phys. Rev. E* 81, 046106. doi:10.1103/physreve.81.046106

Gösgens, M., van der Hofstad, R., and Litvak, N. (2023a). "Correcting for granularity bias in modularity-based community detection methods," in Proceedings Algorithms and Models for the Web Graph: 18th International Workshop, WAW 2023, Toronto, ON, Canada, May 23–26, 2023 (Springer), 1–18.

Gösgens, M., van der Hofstad, R., and Litvak, N. (2023b). The hyperspherical geometry of community detection: modularity as a distance. *J. Mach. Learn. Res.* 24, 1–36.

Gösgens, M. M., Tikhonov, A., and Prokhorenkova, L. (2021). "Systematic analysis of cluster similarity indices: how to validate validation measures," in International Conference on Machine Learning, Cambridge, Massachusetts, USA (PMLR), 3799–3808.

Harper, L. (1967). Stirling behavior is asymptotically normal. *Ann. Math. Statistics* 38, 410–414. doi:10.1214/aoms/1177698956

Holland, P. W., Laskey, K. B., and Leinhardt, S. (1983). Stochastic blockmodels: first steps. *Soc. Netw.* 5, 109–137. doi:10.1016/0378-8733(83)90021-7

Jain, A. K. (2010). Data clustering: 50 years beyond k-means. *Pattern Recognit. Lett.* 31, 651–666. doi:10.1016/j.patrec.2009.09.011

Kamiński, B., Prałat, P., and Théberge, F. (2021). Artificial benchmark for community detection (abcd)—fast random graph model with community structure. *Netw. Sci.* 9, 153–178. doi:10.1017/nws.2020.45

Lambiotte, R., Delvenne, J.-C., and Barahona, M. (2014). Random walks, markov processes and the multiscale modular organization of complex networks. *IEEE Trans. Netw. Sci. Eng.* 1, 76–90. doi:10.1109/tnse.2015.2391998

Lambiotte, R., and Schaub, M. T. (2021). *Modularity and dynamics on complex networks.* Cambridge University Press.

Lancichinetti, A., Fortunato, S., and Radicchi, F. (2008). Benchmark graphs for testing community detection algorithms. *Phys. Rev. E* 78, 046110. doi:10.1103/physreve.78.046110

Lei, Y., Bezdek, J. C., Romano, S., Vinh, N. X., Chan, J., and Bailey, J. (2017). Ground truth bias in external cluster validity indices. *Pattern Recognit.* 65, 58–70. doi:10.1016/j.patcog.2016.12.003

Liu, Z., and Barahona, M. (2018). Geometric multiscale community detection: markov stability and vector partitioning. *J. Complex Netw.* 6, 157–172. doi:10.1093/comnet/cnx028

Meeks, K., and Skerman, F. (2020). The parameterised complexity of computing the maximum modularity of a graph. *Algorithmica* 82, 2174–2199. doi:10.1007/s00453-019-00649-7

Newman, M. E. (2003). Properties of highly clustered networks. *Phys. Rev. E* 68, 026121. doi:10.1103/physreve.68.026121

Newman, M. E. (2006). Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E* 74, 036104. doi:10.1103/physreve.74.036104

Newman, M. E. (2016). Equivalence between modularity optimization and maximum likelihood methods for community detection. *Phys. Rev. E* 94, 052315. doi:10.1103/physreve.94.052315

Newman, M. E., and Girvan, M. (2004). Finding and evaluating community structure in networks. *Phys. Rev. E* 69, 026113. doi:10.1103/physreve.69.026113

Peixoto, T. P. (2019). Bayesian stochastic blockmodeling. *Adv. Netw. Clust. blockmodeling*, 289–332. doi:10.1002/9781119483298.ch11

Peixoto, T. P. (2021). *Descriptive vs. inferential community detection: pitfalls, myths and half-truths arXiv preprint arXiv:2112.00183.*

Peixoto, T. P. (2022). Disentangling homophily, community structure, and triadic closure in networks. *Phys. Rev. X* 12, 011004. doi:10.1103/physrevx.12.011004

Prokhorenkova, L., and Tikhonov, A. (2019). "Community detection through likelihood optimization: in search of a sound model," in The World Wide Web Conference, 1498–1508.

Reichardt, J., and Bornholdt, S. (2006). Statistical mechanics of community detection. *Phys. Rev. E* 74, 016110. doi:10.1103/physreve.74.016110

Rosvall, M., Delvenne, J.-C., Schaub, M. T., and Lambiotte, R. (2019). Different approaches to community detection. *Adv. Netw. Clust. blockmodeling* 105–119, 105–119. doi:10.1002/9781119483298.ch4

Sachkov, V. N. (1997). *Probabilistic methods in combinatorial analysis*, 56. Cambridge University Press.

Traag, V. A., Van Dooren, P., and Nesterov, Y. (2011). Narrow scope for resolution-limit-free community detection. *Phys. Rev. E* 84, 016114. doi:10.1103/physreve.84.016114

Traag, V. A., Waltman, L., and Van Eck, N. J. (2019). From louvain to leiden: guaranteeing well-connected communities. *Sci. Rep.* 9, 5233. doi:10.1038/s41598-019-41695-z

Veldt, N., Gleich, D. F., and Wirth, A. (2018). A correlation clustering framework for community detection. Proceedings of the 2018 World Wide Web Conference, 439–448.

Vinh, N. X., Epps, J., and Bailey, J. (2009). "Information theoretic measures for clusterings comparison: is a correction for chance necessary?," in Proceedings of the 26th annual international conference on machine learning, 1073–1080.

Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics Comput.* 17, 395–416. doi:10.1007/s11222-007-9033-z

Watts, D. J., and Strogatz, S. H. (1998). Collective dynamics of 'small-world'networks. *Nature* 393, 440–442. doi:10.1038/30918

Zhang, L., and Peixoto, T. P. (2020). Statistical inference of assortative community structures. *Phys. Rev. Res.* 2, 043271. doi:10.1103/physrevresearch.2.043271