Journal of Machine Learning Research 24 (2023) 1-36 Submitted 6/22; Revised 11/22; Published 2/23

The Hyperspherical Geometry of Community Detection: Modularity as a Distance

Martijn Gösgens RESEARCH@MARTIJNGOSGENS.NL
Eindhoven University of Technology, Eindhoven, Netherlands

Remco van der Hofstad R.W.V.D.HOFSTAD@TUE.NL
Eindhoven University of Technology, Eindhoven, Netherlands

Nelly Litvak N.LITVAK@UTWENTE.NL
University of Twente, Enschede, Netherlands
Eindhoven University of Technology, Eindhoven, Netherlands

Editor: Edo Airoldi

Abstract

We introduce a metric space of clusterings, where clusterings are described by a binary vector indexed by the vertex-pairs. We extend this geometry to a hypersphere and prove that maximizing modularity is equivalent to minimizing the angular distance to some *modularity vector* over the set of clustering vectors. In that sense, modularity-based community detection methods can be seen as a subclass of a more general class of *projection methods*, which we define as the community detection methods that adhere to the following two-step procedure: first, mapping the network to a point on the hypersphere; second, projecting this point to the set of clustering vectors. We show that this class of projection methods contains many interesting community detection methods. Many of these new methods cannot be described in terms of null models and resolution parameters, as is customary for modularity-based methods. We provide a new characterization of such methods in terms of *meridians* and *latitudes* of the hypersphere. In addition, by relating the modularity resolution parameter to the latitude of the corresponding modularity vector, we obtain a new interpretation of the *resolution limit* that modularity maximization is known to suffer from.

Keywords: Community detection, Louvain algorithm, modularity, clustering, resolution limit

1. Introduction

Complex networks often contain groups of nodes that are more connected internally than externally. In network science, such groups are referred to as *communities*. Community detection is the task of detecting such groups in a network. Numerous algorithms have been developed for this task and its variants (Fortunato, 2010; Fortunato and Hric, 2016; Rosvall et al., 2019). In this work, we consider detecting non-overlapping communities, so that a community structure is represented by a partition of the network nodes into communities. We refer to such partitions as *clusterings*. We define a *hyperspherical geometry* on such clusterings that allows for a natural interpretation of widely-used community detection methods in terms of projections. This geometrical interpretation has several interesting consequences, and opens the door for designing new community detection methods.

In the remainder of this section we explain our main contributions and innovations. In order to do so, we start by introducing basic notations and key notions from the literature that we build upon in this work. We consider a simple undirected graph G with vertex-set $[n] = \{1, \dots, n\}$. For a vertex $i \in [n]$, we denote its degree by $d_i^{(G)}$ and denote the total number of edges by $m_G = \frac{1}{2}\sum_{i\in[n]} d_i^{(G)}$. We use the letters C and T to denote clusterings.

Granularity. We denote the number of clusters that a clustering consists of by $|C|$, while m_C denotes the number of intra-cluster vertex-pairs, which is an integer in between 0 (when $|C| = n$) and the total number of vertex-pairs $N = \binom{n}{2}$ (when $|C| = 1$). The extent to which a clustering resembles the former or latter of these extremes is often referred to as the *granularity* of a clustering: *fine-grained* clusterings consist of many small clusters, while *coarse-grained* clusterings consist of a few large clusters. Granularity is an important notion in this paper, and we use the number of intra-cluster pairs m_C as a measure of the granularity. Alternatively, some works simply quantify granularity by the number of clusters $|C|$, but this has the drawback of being insensitive to the cluster sizes. In other works (Romano et al., 2016; Vinh et al., 2009, 2010), Shannon entropy is used as a measure of granularity. A generalization of Shannon entropy has been shown to be related to m_C (Romano et al., 2016). Simpson's index is another measure of granularity that is widely used in ecology Hunter and Gaston (1988). Simpson's index is related to the number of intra-cluster pairs by $S(C) = m_C/N$.

Validation indices. When the ground truth community structure is available, the performance of a community detection method can be evaluated by comparing the obtained candidate clustering to the ground truth. In this work, we denote the ground truth clustering by T, while we denote other (candidate) clusterings by C. Functions that quantify similarity between C and T are referred to as *validation indices*. There exist many different validation indices (Vinh et al., 2010; Lei et al., 2017), and which one is most suitable depends on the context of the application (Gösgens et al., 2021).

One popular class of validation indices are *pair-counting functions*. Given clusterings T and C, these indices can be expressed as functions of the following four variables: the number of intra-cluster pairs m_T, m_C of T and C, respectively; the number of vertex-pairs m_{TC} that are intra-cluster pairs of both T and C; and the total number of vertex-pairs N. Examples of such pair-counting indices are the Rand index (Rand, 1971), the Jaccard index (Jaccard, 1912), the Hubert Index (Hubert, 1977) and the Correlation Coefficient (Gösgens et al., 2021). Many of these indices are known to be biased towards either clusterings of fine or coarse granularity (Albatineh et al., 2006; Romano et al., 2016; Gösgens et al., 2021; Lei et al., 2017). The Jaccard index is known to be biased towards coarse-grained clusterings while the bias of Rand depends on the granularity of the ground truth (Lei et al., 2017). The Correlation Coefficient does not suffer from this bias and additionally satisfies many other desirable properties (Gösgens et al., 2021). This correlation coefficient is given by

$$\mathrm{CC}(T, C) = \frac{m_{TC}\cdot N - m_T \cdot m_C}{\sqrt{m_T\cdot(N-m_T)\cdot m_C\cdot(N-m_C)}}. \tag{1}$$

One of the desirable properties of the correlation coefficient that we make grateful use of is that it can be transformed to a metric distance by taking its arccosine (Gösgens et al., 2021). This *Correlation Distance* plays a central role in this work and is used as the main validation index throughout this paper.

Modularity. One of the most popular ways to detect communities is by maximizing a quantity called *modularity* (Newman and Girvan, 2004; Reichardt and Bornholdt, 2006). Modularity is based on the paradigm that a graph with community structure has many more intra-communities edges than it would have if one were to rewire the edges at random.

Given a clustering C and a graph G, modularity measures the difference between the number of intra-cluster edges that are present in G and the expected number of intra-cluster edges if G were to be rewired according to some random graph model without community structure. Such a random graph model is usually referred to as a *null model.* While it is theoretically possible to use any random graph model as a null model, the null models that are commonly used in the literature are the Erdős-Rényi (ER) model and Configuration Model (CM). The main result of this paper in Theorem 2 establishes the equivalence between modularity maximization and a nearest-neighbor search in hyperspherical geometry.

The resolution limit. In large graphs, modularity maximization is known to be unable to detect communities below a given size. This *resolution limit* (Fortunato and Barthélemy, 2007; Kumpula et al., 2007; Lancichinetti and Fortunato, 2011) is a serious drawback of modularity-based methods. Basically, it means that modularity maximization is implicitly tuned to detect clusterings of a certain granularity. Modularity is often extended to include a *resolution parameter* to alleviate this problem (Reichardt and Bornholdt, 2006; Kumpula et al., 2007). However, this resolution parameter merely allows one to tune the detection method to a different granularity (Arenas et al., 2008), and does not address the fundamental problem that modularity implicitly has a bias towards clusterings of some granularity (Kumpula et al., 2007; Traag et al., 2011). Furthermore, it is nontrivial to find a resolution parameter value so that modularity optimization is tuned to the desired granularity in a given setting (Arenas et al., 2008; Prokhorenkova, 2019). Recently, modularity optimization has been shown to be equivalent to likelihood maximization (Newman, 2016): for a particular value of the resolution parameter, ER modularity is equivalent to the likelihood function of a Planted Partition Model, while CM modularity is equivalent to the likelihood of a Degree-Corrected Planted Partition Model. While this provides a mathematically principled approach to choosing the resolution parameter (Prokhorenkova and Tikhonov, 2019), it does require making assumptions about the distribution that the network was drawn from and knowledge of its parameters.

The Louvain algorithm. Despite its shortcomings, modularity maximization remains one of the most popular approaches for community detection. In applications, an additional difficulty arises from the fact that modularity maximization is NP-hard (Brandes et al., 2007), which makes its exact maximization infeasible for large graphs. Therefore, practitioners often resort to *approximate* optimization algorithms, the most popular being the so-called Louvain algorithm (Blondel et al., 2008). This algorithm is known to find an approximate modularity maximum in roughly log-linear time. The Louvain algorithm is also known to efficiently optimize other partition-based functions (Prokhorenkova and Tikhonov, 2019).

Other modularity maximization algorithms. Early algorithms optimized modularity by greedily merging communities together (Clauset et al., 2004; Wakita and Tsurumi, 2007). The Louvain algorithm offered a significant improvement: it could achieve higher values of modularity at a significantly lower computational cost (Blondel et al., 2008). Recent improvements to the Louvain algorithm have similar running time, and reach better local maxima (Traag et al., 2019). However, they are not easily modified to optimize the other partition-based functions that we consider in this paper. Algorithms based on Linear Programming (Agarwal and Kempe, 2008) or Simulated Annealing (Guimera and Nunes Amaral, 2005; Lee et al., 2012) are also known to result in better modularity optima than the Louvain algorithm, but these algorithms are usually several orders of magnitude slower. We emphasize that any algorithm that maximizes modularity fits our geometric framework and that our theoretical results apply to each of them. We choose to use the Louvain algorithm throughout the paper because it is the most well-known algorithm, and because it is easily modified to optimize other partition-based functions.

Main innovations in this paper. The central idea of this paper is to describe a *hyperspherical geometry* on the set of clusterings. Our main result is that in terms of this geometry, maximizing modularity is equivalent to minimizing the angular distance to some point on the hypersphere that we call the *modularity vector*, over the set of clusterings. In this geometric viewpoint, modularity can be seen as a class of mappings (parametrized by the null model and resolution parameter) from the set of networks to points on the hypersphere. Then, any algorithm that maximizes modularity (e.g., the well-known Louvain algorithm) can be seen as an (approximate) nearest-neighbor search, finding the clustering that minimizes the distance to the modularity vector. By allowing for different ways of mapping networks to points on the hypersphere, we obtain a general class of community detection methods that we refer to as *projection methods*, as the network is first mapped to a point on the hypersphere and then projected to a clustering vector. We show that many interesting projection methods exist that lie outside the class of modularity-based methods.

Other geometric approaches. We emphasize that in the geometry that we consider here, the graph *as a whole* is represented by a single point in space. This is in contrast to other approaches where individual vertices are mapped to points in space by, e.g., *node2vec* (Grover and Leskovec, 2016) or other node embedding procedures (Cui et al., 2018; Gu et al., 2018). For example, it is known that modularity maximization can be approximated by a vector partitioning problem where each vertex is mapped to a vector in some low-dimensional space (Newman, 2006a; Zhang and Newman, 2015). A similar vector-partitioning approach is employed by Liu and Barahona (2018) to optimize the Markov stability of a random walk defined on the graph. This Markov-stability approach to community detection had previously been related to modularity optimization by Delvenne et al. (2010), who showed that modularity corresponds to a linearization of Markov stability. Schaub et al. (2019) describe a different method for mapping graph vertices to vectors based on random-walk dynamics. Their approach has the advantage that it is also able to detect disassortative communities, in contrast to the previously mentioned Markov-stability approach. The mapping that they employ is related to *diffusion maps* (Coifman et al., 2005; Lafon and Lee, 2006; Fanuel et al., 2022), a non-linear dimensionality-reduction method that embeds vertices to a lower-dimensional Euclidean space via a diffusion process. Finally, spectral clustering (Von Luxburg, 2007; Jain, 2010) is a popular clustering method that maps vertices to points in space based on the leading eigenvectors of the Laplacian matrix and then applies K-means on the resulting set of points. The spatial community detection methods described above have in common that they map graph vertices to vectors in order to cluster these vectors using a spatial method (e.g., K-means or vector partitioning). In contrast, our work aims to better understand modularity maximization, its limitations, and its extensions by considering clustering and graphs as objects residing in the same hyperspherical geometry. This geometry is separate from any geometry of the graph vertices: even if the vertices have a natural embedding in a hyperbolic or any other geometry, then modularity still has the hyperspherical geometrical structure that we describe in this paper.

Data clustering. Note that community detection is closely related to the more general field of data clustering, as we are essentially clustering the network nodes based on network topology. While the focus of the present paper is community detection in networks, the proposed method is not limited to networks alone and is able to cluster *any* pair-wise similarity data, such as the upper or lower triangle of any affinity matrix. For a more general overview of data clustering, we refer to Jain (2010).

Organisation of this paper. This paper is organised as follows. In Section 2, we describe the hyperspherical geometry on the set of clusters. In Section 3, we show that maximizing modularity is equivalent to minimizing the angular distance to a modularity vector. The consequences of this equivalence are discussed in Section 4, while Section 5 explores the projection methods that lie outside the class of modularity-based methods. Finally, in Section 6, we compare a number of projection methods on real-world networks and interpret the results in our geometric framework.

2. The Hyperspherical Geometry of Clustering

For a given clustering C, we define $\boldsymbol{b}(C)$ as the binary N-dimensional vector indexed by the vertex-pairs, where $\boldsymbol{b}(C)_{ij}$ equals $+1$ if i and j are assigned the same cluster in C and -1 otherwise. In this work, binary vectors have entries $+1$ and -1. In some particular cases, we need binary vectors with values 1 and 0, which are obtained by the transformation $\frac{1}{2}(\boldsymbol{b}+\mathbf{1})$, where $\mathbf{1}$ denotes the vector where all entries equal 1. Note that not all ±1 binary vectors correspond to clusterings, since the vector needs to satisfy transitivity. That is, for vertices i, j, k, it must hold that $\boldsymbol{b}(C)_{ik} = 1$ if $\boldsymbol{b}(C)_{ij} = \boldsymbol{b}(C)_{jk} = 1$. Furthermore, every binary vector that does satisfy this transitivity condition corresponds to precisely one clustering C. Importantly, note that all binary vectors have equal (Euclidean) length $\sqrt{N}$, hence, all clustering vectors lie on a hypersphere with radius $\sqrt{N}$ centered around the origin.

For clusterings C_1, C_2, the *angular distance* between their binary vectors is given by

$$d_a(\boldsymbol{b}(C_1), \boldsymbol{b}(C_2)) = \arccos\left(\frac{\langle \boldsymbol{b}(C_1), \boldsymbol{b}(C_2)\rangle}{N}\right), \tag{2}$$

where $\langle \cdot, \cdot \rangle$ denotes the standard inner product. We can easily extend this metric space to the full hypersphere to allow for non-binary vectors. For $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^N$, the angular distance is

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

where $\|\boldsymbol{x}\| = \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle}$ denotes the Euclidean norm. Furthermore, the correlation distance, which is defined as the arccosine of the correlation coefficient as given in (1), can similarly be extended to the full hypersphere by

$$d_{\mathrm{CC}}(\boldsymbol{x}, \boldsymbol{y}) = \arccos\left(\frac{\langle \boldsymbol{x}, \boldsymbol{y}\rangle - \langle \boldsymbol{x}, \boldsymbol{1}\rangle \cdot \langle \boldsymbol{y}, \boldsymbol{1}\rangle / N}{\sqrt{\|\boldsymbol{x}\|^2 - \langle \boldsymbol{x}, \boldsymbol{1}\rangle^2/N}\sqrt{\|\boldsymbol{y}\|^2 - \langle \boldsymbol{y}, \boldsymbol{1}\rangle^2/N}}\right). \tag{3}$$

The above expression is undefined when $\boldsymbol{x}$ or $\boldsymbol{y}$ is a constant vector, i.e., when $\boldsymbol{x} = c\boldsymbol{1}$ for some c. Similarly to Gösgens et al. (2021), we resolve this by the convention that the correlation between a constant and a nonconstant vector is 0, while the correlation between two constant vectors is 1 when their signs are equal and -1 otherwise, leading to correlation distances $\arccos(0) = \frac{1}{2}\pi$, $\arccos(1) = 0$ and $\arccos(-1) = \pi$ for these cases, respectively.

We note that the length of a vector has no meaning in these metric spaces. That is, scaling a vector by a positive scalar does not affect the (angular or correlation) distance to any other vector. This introduces an equivalence relation among vectors, where each equivalence class corresponds to a direction, the representative element of each of these classes is the vector that is on the surface of the hypersphere, given by the *hypersphere projection*

$$\mathcal{H}(\boldsymbol{x}) = \frac{\sqrt{N}}{\|\boldsymbol{x}\|}\boldsymbol{x}.$$

All the definitions in the remainder of this section are invariant under this hypersphere projection.

Hyperspherical geometry. The angular distance defines a geometry on our hypersphere. We now introduce some basic hyperspherical geometry theory that will be needed later on. These hyperspherical results are direct analogues of their spherical counterparts. We refer to Todhunter (1863) and Donnay (2011) for a more complete overview of spherical geometry and trigonometry.

For any two-dimensional plane that contains the origin, its intersection with the hypersphere corresponds to a *great circle* (e.g., the equator of a globe), which is the closest hyperspherical analogue to an infinite straight line in Euclidean geometry. Therefore, we refer to segments of a great circle as *hyperspherical straight lines.* The length of a hyperspherical straight line between two points corresponds to the angular distance between its endpoints. The definition of a hyperspherical angle (the angle that two hyperspherical straight lines make on the hypersphere) is less straightforward: given three distinct points $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$ on the hypersphere, the hyperspherical angle $\angle(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ that the line from $\boldsymbol{x}$ to $\boldsymbol{y}$ makes with the line from $\boldsymbol{z}$ to $\boldsymbol{y}$ at $\boldsymbol{y}$ is given by the *hyperspherical cosine rule*:

$$\cos\angle(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = \frac{\cos d_a(\boldsymbol{x}, \boldsymbol{z}) - \cos d_a(\boldsymbol{x}, \boldsymbol{y}) \cos d_a(\boldsymbol{y}, \boldsymbol{z})}{\sin d_a(\boldsymbol{x}, \boldsymbol{y}) \sin d_a(\boldsymbol{y}, \boldsymbol{z})}. \tag{4}$$

This angle is obtained by projecting $\boldsymbol{x}$ and $\boldsymbol{z}$ on the tangent plane of $\boldsymbol{y}$ on the hypersphere and then simply computing the angle for these projected points via the (Euclidean) cosine rule, as illustrated in Figure 1. Later in this section, we show that the Correlation Distance is a hyperspherical angle.

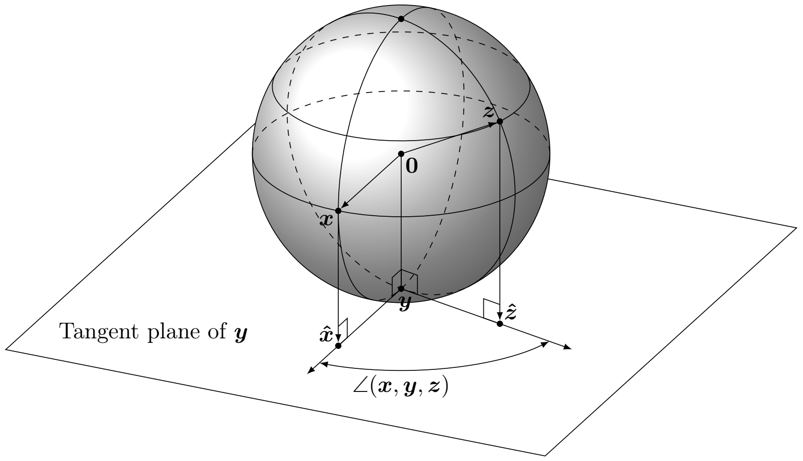


Figure 1: Illustration of the hyperspherical angle $\angle(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ in the three-dimensional subspace spanned by $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$. The solid arcs are on the visible side of the sphere while the dashed arcs are on the back side. The vectors $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{z}}$ are the projections of $\boldsymbol{x}, \boldsymbol{z}$ respectively to the tangent plane of $\boldsymbol{y}$. For $\boldsymbol{y} = -\boldsymbol{1}$, we have $\angle(\boldsymbol{x}, -\boldsymbol{1}, \boldsymbol{z}) = d_{\mathrm{CC}}(\boldsymbol{x}, \boldsymbol{z})$.

Poles. In the metric space described previously, the all-one vector $\boldsymbol{1}$ corresponds to the clustering where all items are put in the same cluster, while the vector $-\boldsymbol{1}$ corresponds to the clustering where each item is placed in its own separate cluster. These two points form opposite *poles* on our hypersphere. Clusterings lying close to $\boldsymbol{1}$ correspond to clusterings of coarse granularity (i.e., consisting of relatively few and large clusters), while clusters close to $-\boldsymbol{1}$ correspond to clusterings of fine granularity. Therefore, we refer to $\boldsymbol{1}$ as the *coarse pole* while we refer to $-\boldsymbol{1}$ as the *fine pole*.

Latitude and granularity. In analogy to the terminology used for globes, we refer to the angular distance to the fine pole as the *latitude* of a vector.[1] We thus define the latitude of a vector $\boldsymbol{x}$ by

$$\ell(\boldsymbol{x}) = d_a(\boldsymbol{x}, -\boldsymbol{1}) = \arccos\Big(\frac{-\langle \boldsymbol{x}, \boldsymbol{1}\rangle}{\sqrt{N} \cdot \|\boldsymbol{x}\|}\Big). \tag{5}$$

If $\boldsymbol{x}$ is a binary vector, then (5) depends only on the number of $+1$'s in $\boldsymbol{x}$. Therefore, for a clustering C with m_C intra-cluster pairs, the latitude is given by

$$\ell(\boldsymbol{b}(C)) = \arccos\left(\frac{-m_C + (N - m_C)}{\sqrt{N} \cdot \sqrt{N}}\right) = \arccos\left(1 - \frac{2m_C}{N}\right). \tag{6}$$

As mentioned previously, m_C is a measure of the granularity of the clustering C. Since $\ell(\boldsymbol{b}(C))$ is a monotonous transformation of m_C, it can equivalently be viewed as a measure of granularity. Furthermore, note that the latitude is related to Simpson's index by $\cos \ell(\boldsymbol{b}(C)) = 1 - 2S(C)$.

We refer to the *equator* as the set of points at equal distance to both poles, that is, with latitude $\pi/2$, or equivalently, the set of vectors $\boldsymbol{x}$ with $\langle \boldsymbol{x}, \boldsymbol{1}\rangle = 0$. This contains the

1. Note that this slightly differs from the definition used in geography, where the latitude is the signed angular distance to the equator.

clustering vectors for clusterings that have an equal number of inter- and intra-cluster pairs. From (6) we see that $\ell(\boldsymbol{b}(C)) > \pi/2$ occurs only when C has more intra-cluster pairs than inter-cluster pairs. Moreover, it can be shown that any clustering with $\ell(\boldsymbol{b}(C)) > \pi/2$ has a cluster of size larger than $n/2$, which is usually not the case when clustering into more than two clusters. This tells us that clusterings generally lie on the fine hemisphere. Furthermore, let $n \to \infty$ and suppose that the expected community size of a randomly chosen vertex is s and does not depend on n, then the latitude of a ground truth community structure shrinks like

$$\arccos\left(1 - 2\frac{s-1}{n-1}\right) = 2\sqrt{\frac{s-1}{n-1}} + o(n^{-1}), \tag{7}$$

so that for large n, clusterings are concentrated around the fine pole.

Meridians and parallels. In the same geographic analogy as above, a *meridian* is a hyperspherical straight line between the poles, while a *parallel* is a hypersurface of constant latitude. Note, however, that in the three-dimensional real world, a parallel is a one-dimensional object while in our case it has dimension $N-2$.

For every vector $\boldsymbol{x}$ that is not a multiple of $\mathbf{1}$, there is precisely one meridian running through its hypersphere projection. Similarly, every vector $\boldsymbol{x}$ lies on exactly one parallel, namely the parallel with latitude $\ell(\boldsymbol{x})$. Therefore, each point on the hypersphere is uniquely defined by a meridian and a latitude. The point that lies on the meridian of $\boldsymbol{x}$ and has latitude λ is equal to the projection of $\boldsymbol{x}$ onto the parallel with latitude λ. This *parallel projection* is given by

$$\mathcal{P}_\lambda(\boldsymbol{x}) = \sin(\lambda)\mathcal{H}\left(\boldsymbol{x} - \frac{\langle \boldsymbol{x}, \mathbf{1}\rangle}{N}\mathbf{1}\right) - \cos(\lambda)\mathbf{1}. \tag{8}$$

Since the equator corresponds to the parallel with latitude $\pi/2$, the projection to the equator is given by $\mathcal{P}_{\pi/2}(\boldsymbol{x})$.

All meridians meet at both poles. Given two meridians that run through $\boldsymbol{x}$ and $\boldsymbol{y}$ respectively, we can compute the hyperspherical angle that they make at the fine pole. We refer to this as the *meridian angle* between $\boldsymbol{x}$ and $\boldsymbol{y}$. For $\boldsymbol{x}, \boldsymbol{y}$ with latitudes in $(0, \pi)$, the meridian angle can be computed via the hyperspherical cosine rule (4) and is given by

$$\text{MeridianAngle}(\boldsymbol{x}, \boldsymbol{y}) = \angle(\boldsymbol{x}, -\mathbf{1}, \boldsymbol{y}) = \arccos\left(\frac{\cos d_a(\boldsymbol{x}, \boldsymbol{y}) - \cos\ell(\boldsymbol{x})\cos\ell(\boldsymbol{y})}{\sin\ell(\boldsymbol{x})\sin\ell(\boldsymbol{y})}\right). \tag{9}$$

Alternatively, this meridian angle can be written as $d_a(\mathcal{P}_{\pi/2}(\boldsymbol{x}), \mathcal{P}_{\pi/2}(\boldsymbol{y}))$. By convention, we define this angle to be zero when both vectors are on the same pole, π when both vectors are on opposite poles and $\pi/2$ when one vector is on a pole while the other is not. We prove the following result:

Theorem 1 *For all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^N$, we have* $\text{MeridianAngle}(\boldsymbol{x}, \boldsymbol{y}) = d_{\text{CC}}(\boldsymbol{x}, \boldsymbol{y})$. *That is, the meridian angle coincides with the correlation distance.*

Proof We prove the theorem by showing that the cosine of the meridian angle in (9) and the cosine of the correlation distance (3) are the same. From (9) we directly get

$$\cos(\text{MeridianAngle}(\boldsymbol{x}, \boldsymbol{y})) = \frac{\cos d_a(\boldsymbol{x}, \boldsymbol{y}) - \cos\ell(\boldsymbol{x}) \cdot \cos\ell(\boldsymbol{y})}{\sqrt{1-\cos^2\ell(\boldsymbol{x})} \cdot \sqrt{1-\cos^2\ell(\boldsymbol{y})}}.$$

Next, we multiply both numerator and denominator by $\|\boldsymbol{x}\| \cdot \|\boldsymbol{y}\|$ to obtain

$$\frac{\|\boldsymbol{x}\| \cdot \|\boldsymbol{y}\| \cos d_a(\boldsymbol{x}, \boldsymbol{y}) - \|\boldsymbol{x}\| \cos \ell(\boldsymbol{x}) \cdot \|\boldsymbol{y}\| \cos \ell(\boldsymbol{y})}{\sqrt{\|\boldsymbol{x}\|^2 - (\|\boldsymbol{x}\| \cos \ell(\boldsymbol{x}))^2} \cdot \sqrt{\|\boldsymbol{y}\|^2 - (\|\boldsymbol{y}\| \cos \ell(\boldsymbol{y}))^2}}.$$

After substituting $\|\boldsymbol{x}\| \cdot \|\boldsymbol{y}\| \cos d_a(\boldsymbol{x}, \boldsymbol{y}) = \langle \boldsymbol{x}, \boldsymbol{y} \rangle$, $\|\boldsymbol{x}\| \cos \ell(\boldsymbol{x}) = -\langle \boldsymbol{x}, \mathbf{1} \rangle / \sqrt{N}$ (see Equation 5) and similarly $\|\boldsymbol{y}\| \cos \ell(\boldsymbol{y}) = -\langle \boldsymbol{y}, \mathbf{1} \rangle / \sqrt{N}$, the resulting expression is the same as the argument of the arccosine in the right-hand side of (3).

We next check the boundary cases, where $\boldsymbol{x}$ or $\boldsymbol{y}$ are poles: The case where both $\boldsymbol{x}$ and $\boldsymbol{y}$ are on the same pole corresponds to comparing two constant vectors of the same sign, which gives correlation 1 and thus results in a correlation distance of 0, which is equal to the meridian angle by our definition. Similarly, when both vectors are on different poles, they are constant vectors of different signs, hence, the correlation coefficient is -1, and the correlation distance again equals the meridian angle π. Finally, when one vector is on the pole and the other is not, (3) gives $d_{\text{CC}}(\boldsymbol{x}, \boldsymbol{y}) = \pi/2$, again in accordance with our definition of the meridian angle. ■

Theorem 1 provides a new geometric interpretation for the correlation distance. Furthermore, since the correlation distance defines a distance on the set of clustering vectors, it can be zero only when clusterings are equal. This implies that all clustering vectors, except the poles, lie on different meridians. To avoid using two different notations for the same quantity, from now on we denote the meridian angle by d_{CC} and refer to it as the correlation distance in the remainder of this paper. As mentioned in Section 1, we use the correlation distance as validation measure in this paper. That is, if T is the ground truth clustering of our network while C is the clustering obtained by some community detection method, then we quantify the performance of that method on that network by $d_{\text{CC}}(\boldsymbol{b}(T), \boldsymbol{b}(C))$.

Other pair-counting distance metrics. The correlation distance is not the only pair-counting validation index that is related to this hyperspherical geometry: the cosine of the angular distance d_a is equal to the Hubert index (Hubert, 1977). This Hubert index is related to the Rand index by $\text{Hubert}(C_1, C_2) = 2\text{Rand}(C_1, C_2) - 1$, while the Rand index is related to the Euclidean distance d_E by

$$\text{Rand}(C_1, C_2) = 1 - \frac{d_E(\boldsymbol{b}(C_1), \boldsymbol{b}(C_2))^2}{4N}.$$

These two relations are relevant because modularity can be related to both Euclidean and angular distance, as we will prove in Section 3.

3. Modularity as a Distance

In the previous section, we have shown how clusterings are mapped to binary vectors in $\mathbb{R}^N$ in a meaningful way by the mapping $\boldsymbol{b}(\cdot)$. In the present section, we discuss how we can similarly map graphs to this same space. This is the next step in our methodology that formalizes community detection methods as algorithms that find a candidate community structure C by minimizing the distance between its binary vector $\boldsymbol{b}(C)$ and the *query vector* $\boldsymbol{q}(G)$ that the graph is mapped to. Here, the function $\boldsymbol{q}(\cdot)$ maps graphs of size n to vectors in $\mathbb{R}^N$, and we refer to $\boldsymbol{q}(\cdot)$ as a *query mapping*. Among others, we will prove that this setup generalizes modularity-based community detection methods.

Query vectors versus affinity matrices. When performing community detection, we want to find $\boldsymbol{q}(G)$ such that the nearest clustering vector is close to the 'true' clustering T. Therefore, $\boldsymbol{q}(G)$ should ideally be chosen in such a way that $\boldsymbol{b}(T)$ is one of its nearest clustering vectors. To achieve this, we wish $\boldsymbol{q}(G)_{ij}$ to be positive whenever ij is an intra-community pair of T and negative otherwise. In that sense, query vectors are related to the notion of an *affinity matrix* that is used throughout the clustering literature (Xu and Wunsch, 2005; Filippone et al., 2008), where the pairwise similarities are summarized by an $n \times n$ symmetric matrix. Existing clustering algorithms use this matrix in various ways to cluster the elements. For example, the popular Spectral Clustering algorithm computes the leading eigenvectors of the affinity matrix and then clusters the items based on these vectors (Filippone et al., 2008). In our context, the $\binom{n}{2} = N$ entries of a pair-wise vector correspond to the entries of the upper (or lower) triangle of this affinity matrix. While for many operations, such as computing eigenvectors, a matrix representation of similarity is natural, we find that in our geometric setting, a vector representation is more convenient because we heavily rely on inner products.

The simplest way to map a graph G to a vector is by a binary *edge-connectivity vector* $\boldsymbol{e}(G)$, where $\boldsymbol{e}(G)_{ij}$ equals $+1$ if vertices i and j are connected by an edge and -1 otherwise. We note that this mapping $\boldsymbol{e}(\cdot)$ forms a bijection between the set of simple undirected graphs of n vertices and the set of binary vectors. However, there are many more ways in which a graph can be mapped to $\mathbb{R}^N$. In Theorem 2 below, we show that maximizing modularity $M^{\mathcal{N}}(C, G; \gamma)$, for any null-model $\mathcal{N}$ and resolution parameter γ, is equivalent to minimizing the distance between the clustering vector $\boldsymbol{b}(C)$ and the *modularity vector* $\boldsymbol{q}(G) = \boldsymbol{q}_M^{\mathcal{N}}(G; \gamma)$ that we define now. Let the N-dimensional vector $\boldsymbol{p}^{\mathcal{N}}(G)$ denote the expected number of edges that each vertex-pair of G has when the edges are rewired according to a null-model $\mathcal{N}$. This vector is non-negative and its entries sum to the total number of edges m_G as the rewiring keeps the total number of edges unchanged. We then define the modularity vector with respect to the null model $\mathcal{N}$ and resolution parameter γ as

$$\boldsymbol{q}_M^{\mathcal{N}}(G; \gamma) = \mathbf{1} + \boldsymbol{e}(G) - 2\gamma \boldsymbol{p}^{\mathcal{N}}(G). \tag{10}$$

When $\gamma = 1$, we may sometimes write $\boldsymbol{q}_M^{\mathcal{N}}(G)$ instead of $\boldsymbol{q}_M^{\mathcal{N}}(G; 1)$ for brevity. For an Erdős-Rényi null model, the expected number of edges is given by $\boldsymbol{p}^{\mathrm{ER}}(G) = \frac{m_G}{N}\mathbf{1}$, while for the Configuration Model, it is given by

$$\boldsymbol{p}^{\mathrm{CM}}(G)_{ij} = \frac{d_i^{(G)} d_j^{(G)}}{2\widetilde{m_G}}, \tag{11}$$

where $\widetilde{m_G}$ is given by

$$\widetilde{m_G} = m_G - \frac{\sum_{i \in [n]} \left(d_i^{(G)}\right)^2}{4 m_G}.$$

In other works, the denominator $2\widetilde{m_G}$ is usually replaced by $2m_G$ for simplicity. In our setting, we need to use $2\widetilde{m_G}$ to ensure that the expectations sum to m_G. Note that in the large-graph limit, both options are equivalent, since $\lim\limits_{n\to\infty} \frac{\widetilde{m_G}}{m_G} = 1$. We now present the main result of this paper:

Theorem 2 *For a null model $\mathcal{N}$ and resolution parameter γ, maximizing modularity $M^{\mathcal{N}}(C, G; \gamma)$ over all clusterings C is equivalent to minimizing either*

(i) the Euclidean distance $d_E(\boldsymbol{b}(C), \boldsymbol{q}_M^{\mathcal{N}}(G; \gamma))$ over all clusterings C; or

(ii) the angular distance $d_a(\boldsymbol{b}(C), \boldsymbol{q}_M^{\mathcal{N}}(G; \gamma))$ over all clusterings C.

Theorem 2 thus gives a geometric interpretation of modularity maximization. We will see that this geometric interpretation is useful. For example, it paves the way for a generalization of modularity-based community detection methods and provides a new interpretation of the resolution parameter γ, as we explain in more detail in Section 4.

To prove Theorem 2, we rely on the following lemma:

Lemma 3 *Minimizing the Euclidean distance to some $\boldsymbol{v}$ over the set of ± 1 binary vectors $\boldsymbol{b}$ is equivalent to minimizing the angular distance between $\boldsymbol{b}$ and $\boldsymbol{v}$.*

Proof The result follows after we relate the square of the Euclidean distance to the cosine of the angular distance, and then use the fact that any binary vector $\boldsymbol{b}$ has length $\sqrt{N}$:

$$\begin{aligned} d_E(\boldsymbol{b}, \boldsymbol{v})^2 &= \|\boldsymbol{b}\|^2 + \|\boldsymbol{v}\|^2 - 2\langle \boldsymbol{b}, \boldsymbol{v}\rangle \\ &= N + \|\boldsymbol{v}\|^2 - 2\|\boldsymbol{v}\|\sqrt{N}\cos d_a(\boldsymbol{b}, \boldsymbol{v}). \end{aligned}$$

Now N and $\|\boldsymbol{v}\|$ are constant w.r.t. $\boldsymbol{b}$ and can thus be ignored. This tells us that minimizing the Euclidean distance is equivalent to maximizing the cosine of the angular distance, or equivalently, minimizing the angular distance. This completes the proof. ■

We are now ready to prove Theorem 2:

Proof of Theorem 2 Modularity can be written as

$$M^{\mathcal{N}}(C, G; \gamma) = \frac{1}{m_G}\langle \tfrac{1}{2}(\mathbf{1} + \boldsymbol{b}(C)), \tfrac{1}{2}(\mathbf{1} + \boldsymbol{e}(G)) - \gamma \boldsymbol{p}^{\mathcal{N}}(G)\rangle. \tag{12}$$

We multiply this by $4m_G$ and subtract $\langle \mathbf{1}, \mathbf{1} + \boldsymbol{e}(G) - 2\gamma \boldsymbol{p}^{\mathcal{N}}(G)\rangle$, both of which are constant w.r.t. C. This yields

$$\langle \boldsymbol{b}(C), \mathbf{1} + \boldsymbol{e}(G) - 2\gamma \boldsymbol{p}^{\mathcal{N}}(G)\rangle = \|\boldsymbol{b}(C)\| \cdot \|\boldsymbol{q}_M^{\mathcal{N}}(G; \gamma)\| \cos d_a(\boldsymbol{b}(C), \boldsymbol{q}_M^{\mathcal{N}}(G; \gamma)).$$

Finally, $\|\boldsymbol{b}(C)\| = \sqrt{N}$ and $\|\boldsymbol{q}_M^{\mathcal{N}}(G; \gamma)\|$ are constant w.r.t. C. From this, we conclude that maximizing modularity is equivalent to maximizing the cosine of the angular distance. Since the arccosine is a strictly decreasing function on $[-1, 1]$, it follows that maximizing modularity is equivalent to minimizing the angular distance. The equivalence to the Euclidean distance follows immediately from Lemma 3. ■

Theorem 2 relates modularity to a hyperspherical geometry as well as a Euclidean geometry, where the latter has one more dimension than the former. This can be explained by the fact that rescaling the query vector does not affect the optimization, as shown in Lemma 3. Therefore, the length of a query vector in Euclidean space has no effect on the resulting candidate clustering. Because of this, we focus on the hyperspherical geometry for the remainder of this work.

We note that Theorem 2 can easily be extended to variants of modularity. For example, modularity can be extended to weighted networks by replacing the positive entries of $\boldsymbol{e}(G)$ by the edge weights and replacing $d_i(G)$ by the sum of edge weights connected to vertex i. Similarly, the equivalence can also be extended to methods with negative edge-weights such as in Traag and Bruggeman (2009). This shows that all modularity-based methods can be formulated in our geometrical framework. However, the class of methods that fit our geometrical framework is not limited to modularity-based methods, as we discuss in Section 5.

4. Consequences for Modularity-Based Methods

In this section, we discuss several consequences of the equivalence proved in Theorem 2. In particular, we provide a geometric interpretation for the Louvain algorithm and the resolution limit.

Louvain as an approximate nearest-neighbor search. Let us denote a clustering vector that minimizes the angular distance to the query vector $\boldsymbol{q}$ by $\mathcal{C}(\boldsymbol{q})$. Possibly, there are multiple clustering vectors at minimal distance to the query vector. In such cases, we choose $\mathcal{C}(\boldsymbol{q})$ arbitrarily (but deterministically) out of them.

Note that $\mathcal{C}(\boldsymbol{q})$ is a *nearest neighbor* of the query vector $\boldsymbol{q}$ among the clustering vectors. However, from Theorem 2 it follows that finding $\mathcal{C}(\boldsymbol{q}_M^{\mathcal{N}}(G;\gamma))$ is equivalent to finding the global modularity maximum, which is known to be NP-hard (Brandes et al., 2007). Therefore, most modularity-maximizing algorithms, such as the popular Louvain algorithm (Blondel et al., 2008), *approximately* maximize modularity, resulting in an *approximate nearest neighbor* of the modularity vector. Hence, the Louvain algorithm can be viewed as an *approximate nearest-neighbor search.*

While the Louvain algorithm was initially introduced for CM-modularity maximization, it is known to be able to efficiently optimize other clustering-functions as well (Traag et al., 2011; Prokhorenkova and Tikhonov, 2019). In this work, we use the Louvain algorithm to minimize the angular distance to a query vector. The computational cost of projecting a query vector $\boldsymbol{q}$ via the Louvain algorithm depends on the query vector and its representation. When $\boldsymbol{q}$ is a modularity vector or any other linear combination of the vectors $\boldsymbol{e}(G)$, $\boldsymbol{1}$, $\boldsymbol{p}^{\mathrm{ER}}(G)$, and $\boldsymbol{p}^{\mathrm{CM}}(G)$ (amongst others), our modified Louvain algorithm will have the same computational complexity as the original Louvain modularity-maximization algorithm, which is known to be log-linear in the number of edges (Sánchez et al., 2016). Other query vectors may result in a higher computational cost, as will be discussed in Section 5.2.

Let $\mathcal{L}(\boldsymbol{q})$ denote the approximate nearest neighbor obtained by applying the Louvain algorithm to a query vector $\boldsymbol{q}$. Then, we expect that $d_a(\mathcal{L}(\boldsymbol{q}),\boldsymbol{q}) \approx d_a(\mathcal{C}(\boldsymbol{q}),\boldsymbol{q})$, though $d_a(\mathcal{L}(\boldsymbol{q}),\boldsymbol{q}) \geq d_a(\mathcal{C}(\boldsymbol{q}),\boldsymbol{q})$ always holds since $\mathcal{C}(\boldsymbol{q})$ is a global minimizer.

Furthermore, because the Louvain algorithm is a greedy algorithm that is initialized at the clustering corresponding to the fine pole $-\boldsymbol{1}$, it follows that $\mathcal{L}(\boldsymbol{q})$ cannot be further from $\boldsymbol{q}$ than $-\boldsymbol{1}$, i.e. $d_a(\mathcal{L}(\boldsymbol{q}),\boldsymbol{q}) \leq d_a(-\boldsymbol{1},\boldsymbol{q}) = \ell(\boldsymbol{q})$. Combining this, we get

$$d_a(\mathcal{L}(\boldsymbol{q}),\boldsymbol{q}) \in [d_a(\mathcal{C}(\boldsymbol{q}),\boldsymbol{q}),\ell(\boldsymbol{q})]. \tag{13}$$

Projection methods. A mapping $\mathcal{A}$ is a *projection* if $\mathcal{A}(\mathcal{A}(\boldsymbol{x})) = \mathcal{A}(\boldsymbol{x})$ holds for all $\boldsymbol{x} \in \mathbb{R}^N$. It can easily be seen that the hypersphere projection $\mathcal{H}$ and the parallel projection $\mathcal{P}_\lambda$, for any $\lambda \in [0, \pi]$, are indeed projections. Also, it holds that $\mathcal{C}$ is a projection onto the set of clustering vectors, as $\mathcal{C}(\boldsymbol{b}(C)) = \boldsymbol{b}(C)$ holds for all clustering vectors. In Appendix A, we furthermore prove that the Louvain algorithm is also a projection. Therefore, modularity maximization using the Louvain algorithm belongs to the broader class of *projection methods* that we define now:

Definition 4 *A community detection method is a* projection method *if it can be described by the following two-step approach: 1) the graph is first mapped to a query vector; and 2) the query vector is projected to the set of clustering vectors.*

In Section 5, we discuss various interesting options for the first step, while we use the Louvain algorithm for the second step.

Resolution and latitude. The resolution parameter in the modularity function is closely related to the latitude of the corresponding modularity vector, as described by the following lemma:

Lemma 5 *The line $(\mathcal{H}(\boldsymbol{q}_M^{\mathcal{N}}(G;\gamma)))_{\gamma \geq 0}$ is the (hyperspherical) straight line that starts from $\mathcal{H}(\mathbf{1} + \boldsymbol{e}(G))$, intersects the equator at $\gamma = 1$ and ends in $\mathcal{H}(-\boldsymbol{p}^{\mathcal{N}}(G))$. The resolution parameter relates to the latitude as*

$$\ell(\boldsymbol{q}_M^{\mathcal{N}}(G;\gamma)) = \arccos\left(\frac{(\gamma-1)m_G}{\sqrt{N}\cdot\sqrt{(1-\gamma)m_G + \gamma^2\left\|\boldsymbol{p}^{\mathcal{N}}(G)\right\|^2 - \gamma\langle \boldsymbol{p}^{\mathcal{N}}(G), \boldsymbol{e}(G)\rangle}}\right). \tag{14}$$

Proof For $\gamma = 0$, the modularity vector is given by $\boldsymbol{q}_M^{\mathcal{N}}(G;0) = \mathbf{1} + \boldsymbol{e}(G)$, corresponding to the starting point of the line. The endpoint is obtained by

$$\lim_{\gamma\to\infty} \mathcal{H}(\boldsymbol{q}_M^{\mathcal{N}}(G;\gamma)) = \lim_{\gamma\to\infty} \sqrt{N}\frac{\mathbf{1} + \boldsymbol{e}(G) - 2\gamma\boldsymbol{p}^{\mathcal{N}}(G)}{\|\mathbf{1} + \boldsymbol{e}(G) - 2\gamma\boldsymbol{p}^{\mathcal{N}}(G\|} = -\sqrt{N}\frac{\boldsymbol{p}^{\mathcal{N}}(G)}{\|\boldsymbol{p}^{\mathcal{N}}(G)\|} = \mathcal{H}(-\boldsymbol{p}^{\mathcal{N}}(G)).$$

To prove that the curve $\gamma \mapsto \mathcal{H}(\boldsymbol{q}_M^{\mathcal{N}}(G;\gamma))$ is a hyperspherical straight line, we must show that it is a segment of a great circle. Note that, according to (10), the line $(\boldsymbol{q}_M^{\mathcal{N}}(G;\gamma))_{\gamma\geq 0}$ is contained in the 2-dimensional plane that is defined by the three vectors $\boldsymbol{q}_M^{\mathcal{N}}(G;0)$, $\boldsymbol{q}_M^{\mathcal{N}}(G;1)$ and the origin. The intersection between this plane and the hypersphere defines a great circle. Since the projection $(\mathcal{H}(\boldsymbol{q}_M^{\mathcal{N}}(G;\gamma)))_{\gamma\geq 0}$ is contained in the same plane and lies on the hypersphere, it must be a subset of the same great circle. Therefore, $(\mathcal{H}(\boldsymbol{q}_M^{\mathcal{N}}(G;\gamma)))_{\gamma\geq 0}$ is a segment of a great circle, so that it is a hyperspherical straight line by definition. Finally, (14) is obtained by substituting the modularity vector, given by (10), into the latitude ℓ as given by (5), and using $\langle \boldsymbol{p}^{\mathcal{N}}(G), \mathbf{1}\rangle = m_G$. ■

In Lemma 5, we restrict to $\gamma \geq 0$. While modularity could also be defined for *negative* resolution parameter values, the optimization of modularity for $\gamma < 0$ always leads to clustering all items in the same cluster (i.e., the coarse pole). Lemma 5 can easily be extended to take these negative resolution parameters to show that $(\mathcal{H}(\boldsymbol{q}_M^{\mathcal{N}}(G;\gamma)))_{\gamma\in\mathbb{R}}$ corresponds to

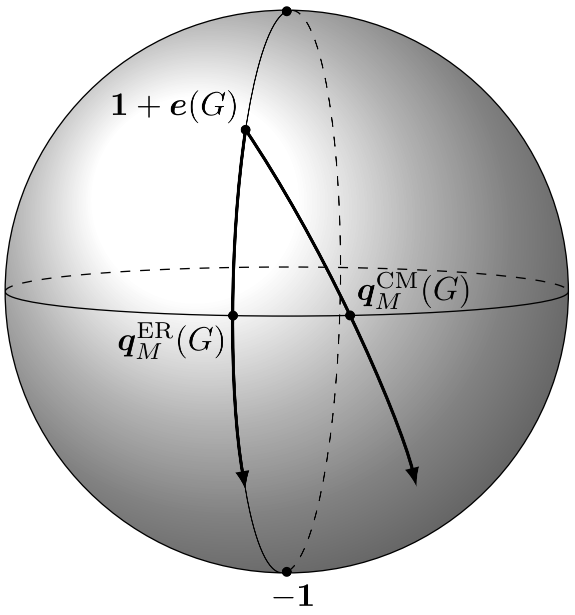

$\mathbf{1}+\boldsymbol{e}(G)$
$\boldsymbol{q}_M^{\mathrm{CM}}(G)$
$\boldsymbol{q}_M^{\mathrm{ER}}(G)$
$-\mathbf{1}$

Figure 2: Illustration of the hyperspherical lines formed by varying the latitude of the modularity vector for a null model. The ER-modularity vectors lie on a single meridian, in contrast to the CM-modularity vectors.

the unique hyperspherical straight line from $\mathcal{H}(\boldsymbol{p}^{\mathcal{N}}(G))$ to $\mathcal{H}(-\boldsymbol{p}^{\mathcal{N}}(G))$ that passes through $\mathcal{H}(\mathbf{1}+\boldsymbol{e}(G))$.

Figure 2 provides a three-dimensional illustration of what these modularity lines may look like. For the ER null-model, we have $\mathcal{H}(\boldsymbol{p}^{\mathcal{N}}(G))=\mathbf{1}$, so Lemma 5 tells us that the line $\mathcal{H}(\boldsymbol{q}_M^{\mathrm{ER}}(G;\gamma))_{\gamma\geq 0}$ lies on a single meridian, ranging from $\mathcal{H}(\mathbf{1}+\boldsymbol{e}(G))$ to $-\mathbf{1}$. The latitudes of vectors on this meridian are given by

$$\tan \ell(\boldsymbol{q}_M^{\mathrm{ER}}(G;\gamma)) = \frac{\sqrt{\frac{N-m_G}{m_G}}}{\gamma-1},$$

as is derived in Appendix A. For other null models such as CM, this line does not correspond to a meridian, but passes through a range of meridians.

Already from the example of the ER and CM null models, we see that the latitude of the modularity vector, described by (14) in Lemma 5, depends on the particular null model. In other words, two modularity vectors with the same resolution parameter but different null models have different latitudes. The advantage of latitude is that it puts all these modularity and clustering vectors on the same scale, allowing for a comparison of modularity-based methods of different null models.

The hyperspherical interpretation of the resolution limit. A well-known limitation of modularity-maximizing methods is the so-called *resolution limit* (Fortunato and Barthélemy, 2007). The easiest example to demonstrate this resolution limit is by a ring of cliques (Fortunato and Barthélemy, 2007): consider a graph $G_{k,s}$ that is given by a ring of k cliques each consisting of s vertices and let each pair of neighboring cliques be connected by a single edge. Such a graph has a natural clustering $T_{k,s}$ into its k cliques. It can be shown that for any $s > 2$ and sufficiently high k, a clustering that merges two neighboring cliques has a higher modularity value, which is clearly undesirable. While this result was initially only given for the CM null model and $\gamma = 1$, it easily generalizes to other null models, resolution parameter values and graph designs (Fortunato and Barthélemy, 2007; Traag et al., 2011).

The general problem is that in large sparse graphs, the expected number of edges between small communities is so negligibly small that the existence of *any* edge between them is enough to result in a modularity-increase for merging these communities. In our geometric framework, we will show that this resolution limit can be characterized as a discrepancy between the latitude of the query vector and the latitude of the ground truth clustering vector. The inverse triangle inequality allows us to bound the angular distance by

$$d_a(\boldsymbol{b}(T), \boldsymbol{q}_M^{\mathcal{N}}(G;\gamma)) \geq |d_a(-\mathbf{1}, \boldsymbol{q}_M^{\mathcal{N}}(G;\gamma)) - d_a(-\mathbf{1}, \boldsymbol{b}(T))| = |\ell(\boldsymbol{q}_M^{\mathcal{N}}(G;\gamma)) - \ell(\boldsymbol{b}(T))|.$$

Note that for $\gamma = 1$, the modularity vector has latitude $\pi/2$ while the latitude of the ground truth clustering vector decreases roughly as $\mathcal{O}(n^{-1/2})$, as shown in (7). Therefore, $d_a(\boldsymbol{b}(T), \boldsymbol{q}_M^{\mathcal{N}}(G;1)) \approx \pi/2$ for large graphs. This means that on this hypersphere, the vectors $\boldsymbol{b}(T)$ and $\boldsymbol{q}_M^{\mathcal{N}}(G;\gamma)$ are, in a way, *half a world apart.* Since modularity maximization returns a clustering C with $\boldsymbol{b}(C)$ near $\boldsymbol{q}_M^{\mathcal{N}}(G;\gamma)$, this clustering C will likely be far from the ground truth clustering T. This characterizes the resolution limit as a discrepancy between the latitudes of the modularity vector and ground truth clustering vector.

However, using a different choice for the query vector could avoid such a resolution limit altogether: for the particular example of the ring of cliques $G_{k,s}$, the simple query mapping $\boldsymbol{q}(G) = \boldsymbol{e}(G)$ would have angular distance

$$d_a(\boldsymbol{b}(T), \boldsymbol{e}(G_{k,s})) = \arccos\left(1 - 2\frac{k}{N}\right) \approx 2\sqrt{k/\binom{k\cdot s}{2}} = \mathcal{O}(k^{-1/2}),$$

for $k \to \infty$. Therefore, this query mapping overcomes the resolution limit and is thus better than the modularity query mapping for this particular graph. We emphasize that the choice of the query vector should depend on characteristics of the graph and community structure: for more realistic networks with larger and less dense communities, the query mapping $\boldsymbol{q}(G) = \boldsymbol{e}(G)$ performs poorly as its latitude is small compared to that of the ground truth clustering. We note that $\boldsymbol{e}(G)$ does lie on the same meridian as the ER-modularity vector[2] while its latitude decreases as $\mathcal{O}(n^{-1/2})$, which suggests that adjusting the latitude of a query vector is a suitable way of overcoming such resolution limits.

Based on the discussion above, we conclude that the latitude of the query vector is a more informative quantity than the resolution parameter. Indeed, the latitude has the advantage that it is defined for any query and clustering vector, while the resolution parameter only tells us something about the specific query vector $\boldsymbol{q}_M^{\mathcal{N}}(G;\gamma)$. Therefore, we suggest to use the latitude of the query vector, rather than the resolution parameter, as the quantity that regulates the granularity of the candidate clustering.

2. More precisely, $\boldsymbol{e}(G) = \boldsymbol{q}_M^{\mathrm{ER}}\left(G; \frac{N}{2m_G}\right)$.

5. Beyond Modularity-Based Methods

As shown in Section 3, modularity merely corresponds to a subclass of possible query mappings. In this section, we discuss useful query mappings that do not fall under the classical formulation in terms of null models and resolution parameters. We will argue that for this wider class of projection methods, it is unnatural to think in terms of null models and resolution parameters, but that these methods instead are better characterized in terms of the meridians and latitudes of the corresponding query vectors.

5.1 Beyond Null Models

From Lemma 5, we learned that, for different null models, modularity corresponds to different lines on the hypersphere, each starting from $\mathcal{H}(\mathbf{1}+\boldsymbol{e}(G))$ and crossing the equator at different points. The line of ER modularity corresponds to the meridian of $\boldsymbol{e}(G)$, while the line of CM modularity passes through a range of meridians and intersects the ER meridian at $\mathcal{H}(\mathbf{1}+\boldsymbol{e}(G))$, as illustrated by Figure 2. Therefore, linear combinations of these two modularity vectors define a two-dimensional subspace of the hypersphere. We observe that the best-performing query vectors among this hyperplane generally do not lie on one of those two modularity lines. As an example, we take the well-known Karate network (Zachary, 1977) and compute the performance (in terms of the correlation $\mathrm{CC}(T,C)$, as defined in Equation 1) for a range of query vectors in this two-dimensional set. The results are shown in Figure 3. We see that both ER and CM modularity perform reasonably well, but are outperformed by other query vectors. We further see that there is a region of query vectors above the CM-modularity line for which the true community structure is recovered. Figure 4 shows the same experiment for the Football network (Girvan and Newman, 2002), where each vertex represents an American college football team, each edge represents a match played between the teams, and each community corresponds to a so-called conference. Again, the ER- and CM-modularity lines are mostly outside of the region of best performing queries.

Note the sharp transition in Figure 3, where the region of 'good' query vectors directly neighbors the region of poorly performing query vectors, i.e., the query vectors that lead to correlation 0. This is easily explained by the fact that the perfectly performing query vectors yield candidates consisting of two communities. Therefore, increasing the query latitude eventually leads to these two communities merging into the clustering consisting of a single community, which has zero correlation to the true clustering. This behavior is different when the ground truth clustering has more than two communities. Indeed, in Figure 4, the best-performing query vectors for the Football networks do not lie right next to poorly-performing query vectors since the ground truth clustering consists of 12 communities.

Null models. Since the query vectors from Figures 3 and 4 are linear combinations of the ER- and CM-modularity vector, one may be tempted to think that these query vectors themselves correspond to modularity vectors with some null model that is a mixture of ER and CM. However, this is generally not the case. Specifically, although for each of these query vectors $\boldsymbol{q}$ there exist γ, c_1, c_2 such that

$$\boldsymbol{q} = \mathbf{1}+\boldsymbol{e}(G) - 2\gamma(c_1\boldsymbol{p}^{\mathrm{ER}}(G) + c_2\boldsymbol{p}^{\mathrm{CM}}(G)),$$

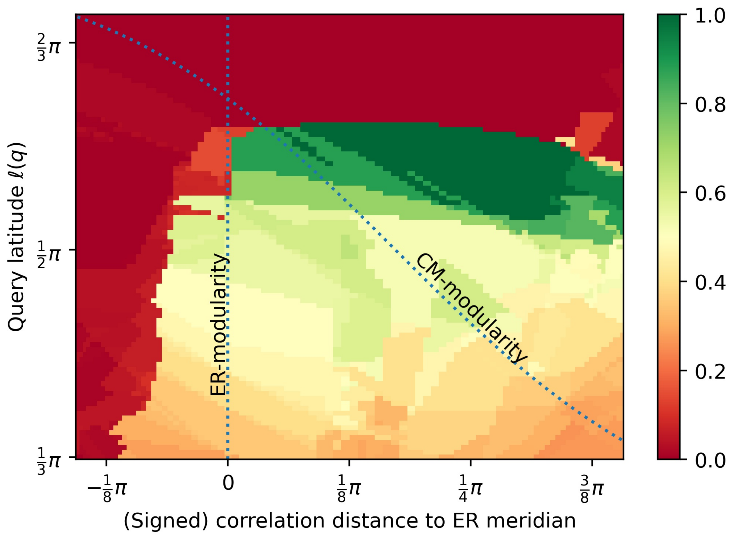


Figure 3: Heatmap of the correlation between the ground truth and the candidate clusterings for the Karate network. We take query vectors from the meridians that $\left(\boldsymbol{q}_M^{\mathrm{CM}}(G;\gamma)\right)_{\gamma\in[-1.5,2]}$ runs through and vary the query latitude between $\frac{1}{3}\pi$ and $\frac{2}{3}\pi$. The horizontal coordinates are given by $\operatorname{sgn}(\gamma)\cdot d_{\mathrm{CC}}(\boldsymbol{q},\boldsymbol{e}(G))$, i.e. the (signed) correlation distance to the ER meridian.

it is not generally the case that c_1 and c_2 are *positive*. Because of this, the 'expected number of edges' may be negative for certain vertex-pairs. Therefore, this linear combination does not fit the requirements corresponding to a null model. For example, each of the perfectly-performing query vectors of Figure 3 corresponds to negative values of c_1. In Appendix B, we show that for another network, the best-performing query vectors correspond to negative values of c_2. For this reason, it is not natural to think of these query vectors as corresponding to some null model. Instead, they may simply be viewed as points on the hypersphere that have a good relative position to the ground truth clustering vector.

5.2 Query Mappings Based on Common Neighbors

Similarly to how we expressed the connectivity of a graph G by the edge-connectivity query vector $\boldsymbol{e}(G)$ in Section 3, we can express the number of common neighbors between each pair of vertices in terms of a vector as well. We denote this vector by $\boldsymbol{w}(G)$ and refer to it as the *wedge vector*, because the entry $\boldsymbol{w}(G)_{ij}$ corresponds to the number of distinct *wedges* (paths of length 2) that have the vertices i and j as endpoints. Intuitively, $\boldsymbol{w}(G)$ is closely related to the community structure of G because disconnected vertices i and j of the same community may have many common neighbors, which makes $\boldsymbol{w}(G)_{ij}$ high as well, and thus $\boldsymbol{b}(C)$ tends to be closer to $\boldsymbol{w}(G)$ when clustering C puts i and j in the same community.

To provide further motivation for the claim that this wedge vector is of interest, we relate it to the *global clustering coefficient* of a graph G, which is defined as thrice the number of triangles (i.e., complete subgraphs of size 3) divided by the number of wedges (i.e., pairs of adjacent edges). Note that each triangle consists of three wedges, so that this quantity equals 1 whenever the graph consists of disconnected cliques, and 0 when the graph is acyclic

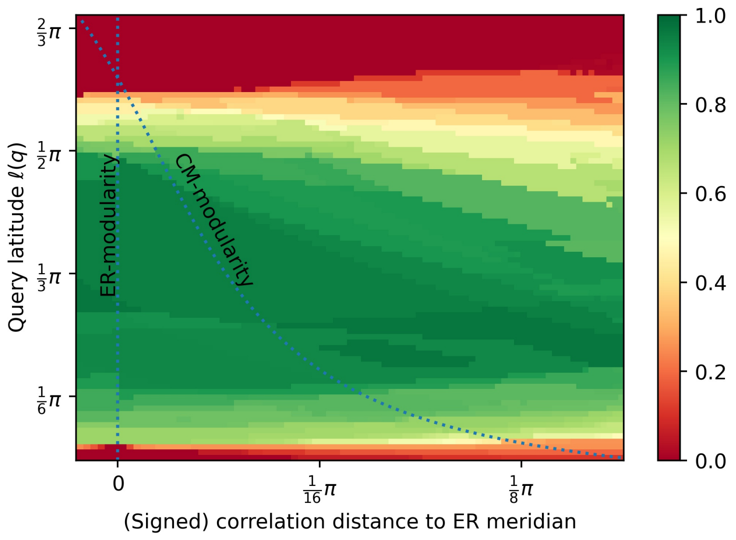


Figure 4: Heatmap of the correlation between the ground truth and the candidate clusterings for the Football network. We take query vectors from the meridians that $\left(\boldsymbol{q}_M^{\mathrm{CM}}(G;\gamma)\right)_{\gamma\in[-1,14]}$ runs through. The horizontal coordinates are given by $\mathrm{sgn}(\gamma)\cdot d_{\mathrm{CC}}(\boldsymbol{q},\boldsymbol{e}(G))$, i.e. the (signed) correlation distance to the ER meridian.

or all cycles have length at least 4. Thus, this clustering coefficient quantifies how much the graph *resembles a clustering*. We prove the following result:

Lemma 6 *The global clustering coefficient of a graph G is given by*

$$\mathrm{GlobalClustering}(G)=\frac{1}{2}\left(1-\frac{\cos d_a(\boldsymbol{e}(G),\boldsymbol{w}(G))}{\cos\ell(\boldsymbol{w}(G))}\right).$$

Proof The global clustering coefficient is given by the fraction of wedges that are closed. Since each triangle consists of three closed wedges, this coefficient is given by thrice the number of triangles divided by the number of wedges. If there is an edge between i and j, then this edge ij is part of exactly $\boldsymbol{w}(G)_{ij}$ triangles. Summing $\boldsymbol{w}(G)_{ij}$ over all edges thus gives thrice the number of triangles, as each triangle contains three edges. In vector notation, this is given by $\langle\frac{1}{2}(\boldsymbol{e}(G)+\mathbf{1}),\boldsymbol{w}(G)\rangle$, where $(\boldsymbol{e}(G)+\mathbf{1})/2$ is the $\{0,1\}$-binary vector edge-connectivity vector. The total number of wedges is given by $\langle\mathbf{1},\boldsymbol{w}(G)\rangle$. Combined, this allows us to write the clustering coefficient as

$$\mathrm{GlobalClustering}(G)=\frac{\langle\frac{1}{2}(\boldsymbol{e}(G)+\mathbf{1}),\boldsymbol{w}(G)\rangle}{\langle\mathbf{1},\boldsymbol{w}(G)\rangle}=\frac{1}{2}\left(1+\frac{\langle\boldsymbol{e}(G),\boldsymbol{w}(G)\rangle}{\langle\mathbf{1},\boldsymbol{w}(G)\rangle}\right).$$

Finally, from the definitions of the angular distance and latitude, as given by (2) and (5), it follows that

$$\frac{\langle\boldsymbol{e}(G),\boldsymbol{w}(G)\rangle}{\langle\mathbf{1},\boldsymbol{w}(G)\rangle}=-\frac{\cos d_a(\boldsymbol{e}(G),\boldsymbol{w}(G))}{\cos\ell(\boldsymbol{w}(G))},$$

which completes the proof. ■

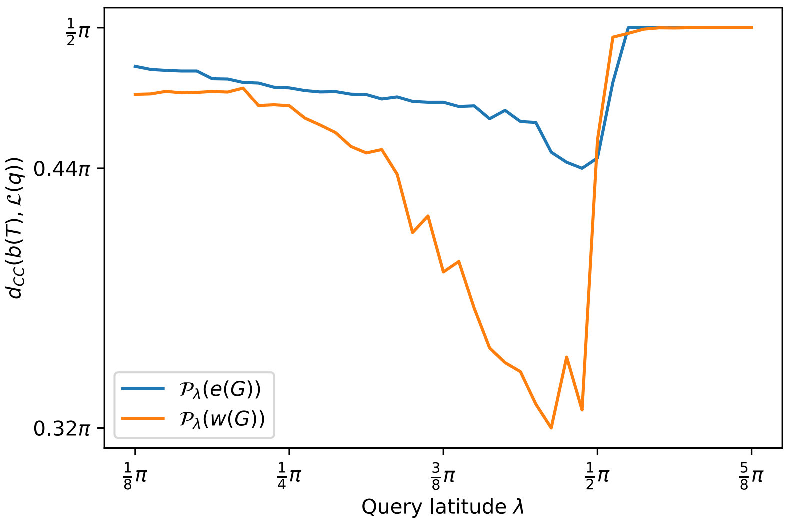


Figure 5: Comparing the performances of query vectors based on the wedge vector $\boldsymbol{w}(G)$ and edge vector $\boldsymbol{e}(G)$ for a PPM consisting of 10 communities, each of size 100, with expected inter- and intra-community degree 5.

Since $\boldsymbol{w}(G)$ is non-negative, by (5), its latitude is at least $\pi/2$, so we always have $\cos\ell(\boldsymbol{w}(G)) \leq 0$. Then Lemma 6 tells us that the global clustering coefficient of a graph G is high whenever the angular distance between $\boldsymbol{w}(G)$ and $\boldsymbol{e}(G)$ is low. This relation to the global clustering coefficient shows that the wedge vector indeed contains relevant information about our graph. However, if we would directly use the vector $\boldsymbol{w}(G)$ as a query vector, we would run into a problem because $\boldsymbol{w}(G)$ has a latitude at least $\pi/2$, and usually much higher than $\pi/2$, so the resulting Louvain candidate $\mathcal{L}(\boldsymbol{w}(G))$ groups all vertices into the same community.

The most straightforward option is to project $\boldsymbol{w}(G)$ to the equator, resulting in the query mapping $\boldsymbol{q}_w(G) = \mathcal{P}_{\pi/2}(\boldsymbol{w}(G))$. The motivation for this choice is that, by Lemma 5, the modularity vector has latitude $\pi/2$ for the default resolution parameter value $\gamma = 1$. Thus, the projection to the equator is comparable to the most common form of modularity. Being on the equator implies that $\langle \boldsymbol{q}_w(G), \mathbf{1}\rangle = 0$. Then, the positive entries of $\boldsymbol{q}_w(G)$ correspond to vertex-pairs that have more common neighbors than the graph average (over the vertex pairs), while negative entries correspond to vertex-pairs that have less common neighbors than average. An approximate nearest neighbor of $\boldsymbol{q}_w(G)$ then corresponds to a clustering where many of the intra-cluster pairs have more common neighbors than average, while many of the inter-cluster pairs have fewer common neighbors than average. This wedge vector can also be projected to other latitudes to obtain a clustering of the desired granularity.

Detecting large communities. We observe that the wedge vector is especially useful as query vector when the community sizes are significantly larger than the average degrees, or equivalently, when $\ell(\boldsymbol{b}(T))$ is significantly larger than $\ell(\boldsymbol{e}(G))$. In Figure 5 we compare wedge-based query vectors to edge-based query vectors on a Planted Partition Model[3] (PPM) with 10 communities each of size 100 and average degree 10, which gives $\ell(\boldsymbol{e}(G)) = 0.064\pi$ and $\ell(\boldsymbol{b}(T)) = 0.204\pi$. We see that, for the right query latitude, the wedge-based query vectors result in candidates with correlation distance 0.32π to the ground truth clustering, while the edge-based query vectors only result in candidates with correlation distance 0.44π, which translate to correlation coefficients of 0.54 and 0.20 respectively. This significant difference in performance may be explained by the fact that the edge vector only connects each vertex to 5 of its 99 community members on average, while the wedge vector connects each vector to approximately 25 community-members.

Computational cost. A practical downside of projection methods based on wedge vectors compared to edge vectors is the higher computational cost. The Louvain algorithm is known to run in roughly log-linear time in terms of the number of edges. When a network is sparse, meaning that the number of edges is of the same order as the number of vertices, this results in computation times that are log-linear in the number of vertices. Similarly, when applying the Louvain algorithm to the wedge vector, the computational time is roughly log-linear in the number of vertex-pairs that have wedges between them. This can be upper-bounded by the total number of wedges, which is given by

$$\langle \boldsymbol{w}(G), \mathbf{1}\rangle = \sum_{i\in[n]} \binom{d_i^{(G)}}{2}.$$

If the degree distribution of the network has a finite second moment, then the expected value of this sum is $\mathcal{O}(n)$, so that the Louvain algorithm will run in log-linear time. However, many real-world networks are known to be scale-free (Barabási, 2013; Fortunato, 2010; Stegehuis et al., 2016; Voitalov et al., 2019), meaning that their degree distribution has a finite mean, but infinite variance. In such cases, the total number of wedges is $\mathcal{O}(n^{2/(\tau-1)})$, where $\tau \in (2, 3)$ is the power-law exponent of the degree distribution,[4] leading to higher computational costs for projection methods based on wedge vectors. We demonstrate this by the experiment reported in Table 1. In this experiment, we generate random networks using the LFR benchmark (Lancichinetti et al., 2008), ranging between 1,000 and 100,000 vertices. These networks have pre-defined communities, and power-law degree distributions with exponents $\tau = 2.9$ and $\tau = 4$. We keep all other parameters (including the average degree) fixed. The number of wedges grows linearly when $\tau = 4$, and faster than linearly when $\tau < 3$. We then apply the projection method using the wedge vector $\mathcal{P}_{\pi/2}(\boldsymbol{w}(G))$ as query vector. Table 1 shows that the computation time for the wedge vector grows much faster for $\tau = 2.9$ than for the exponent $\tau = 4$. This confirms the analytical results that a degree distribution with a heavier tail results in slower computations for wedge-based projection methods. Table 1 also shows the computation time for the query mapping $\boldsymbol{q}_M^{\mathrm{CM}}(G; 1)$, which is based on edges instead of wedges. We see that for this query mapping, the difference in computation times between the two power-law exponent values is much smaller, which is explained by the fact that the expected number of edges is equal in both graphs.

3. That is, a Stochastic Block Model where the block matrix has diagonal entries p_{in} and off-diagonal entries p_{out}.
4. That is, the proportion of vertices having degree k is approximately proportional to $k^{-\tau}$.

	LFR $\tau = 2.9$		LFR $\tau = 4$	
n	$\mathcal{P}_{\pi/2}(\boldsymbol{w}(G))$	$\boldsymbol{q}_M^{\mathrm{CM}}(G;1)$	$\mathcal{P}_{\pi/2}(\boldsymbol{w}(G))$	$\boldsymbol{q}_M^{\mathrm{CM}}(G;1)$
1,000	112s	9s	55s	11s
2,000	269s	22s	37s	7s
5,000	794s	82s	156s	24s
10,000	2,314s	194s	349s	60s
20,000	2,080s	186s	824s	180s
50,000	8,909s	835s	2,677s	701s
100,000	25,794s	2,874s	5,685s	3,092s

Table 1: Computation times for the projection method with query mappings $\mathcal{P}_{\pi/2}(\boldsymbol{w}(G))$ and $\boldsymbol{q}_M^{\mathrm{CM}}(G;1)$, respectively, on LFR networks with power-law exponents $\tau = 2.9$ and $\tau = 4$.

These experiments also demonstrate that the projection method with the Louvain algorithm is computationally feasible in large-scale networks. We emphasize that we have implemented our modification of Louvain in Python, which is significantly slower than other programming languages (e.g., C++) for such tasks. For example, the *NetworKit* (Staudt et al., 2016) implementation of the Louvain algorithm for maximizing the Newman-Girvan modularity on the LFR network of 100,000 vertices with $\tau = 4$ takes only 0.46 seconds. This task is identical to the projection method with query vector $\boldsymbol{q}_M^{\mathrm{CM}}(G;1)$), which takes 3092 seconds with our slow Python implementation. We expect that implementing the projection method for the wedge vector in a similar optimized way would result in a similar improvement in running time.

6. Empirical Analysis of Projection Methods

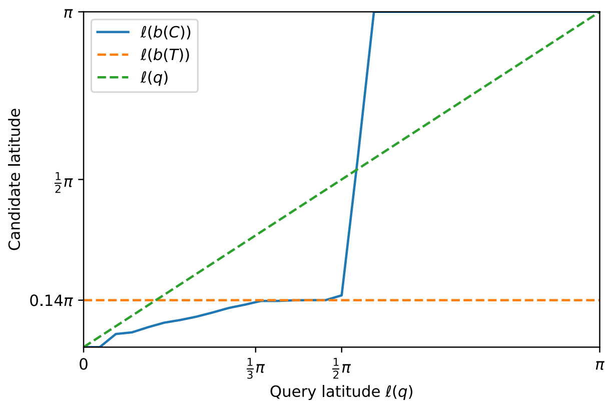
In this section, we perform more experiments to demonstrate how the geometric results reported in this paper help to interpret and understand outcomes of the projection method for different query mappings. In Section 6.1, we illustrate some phenomena that we empirically observe for many networks and query mappings, while in Section 6.2, we compare the performance of different query mappings on several real-world networks.

The code that was used to perform the experiments and generate the figures of this paper is available on GitHub.[5] Amongst others, this repository contains a Python implementation of our modification of the Louvain algorithm. This implementation is able to compute the Louvain projection for a range of query vectors, including all query vectors that are linear combinations of $\mathbf{1}$, $\boldsymbol{e}(G)$, $\boldsymbol{w}(G)$, $\boldsymbol{q}_M^{\mathrm{CM}}(G;\gamma)$, and $\boldsymbol{q}_M^{\mathrm{ER}}(G;\gamma)$.

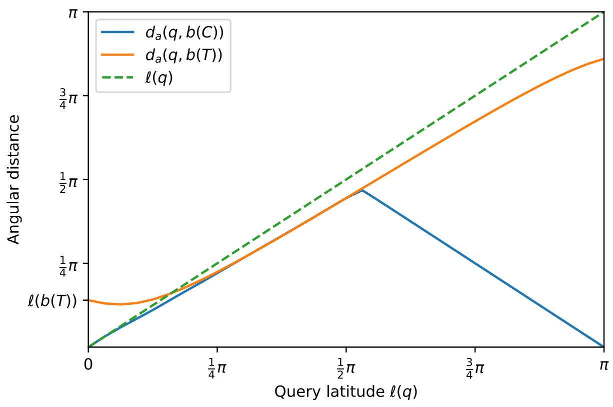
6.1 The Louvain Candidate

In this section, we describe how several geometric quantities are affected when the query latitude is varied. The relevant quantities are summarized in Table 2. For an illustrational experiment, we use query vectors on the ER meridian to detect communities in a Planted Partition Model (PPM). The ground truth clustering consists of 20 communities, each of size 20, while the mean intra- and inter-community degrees are 6 and 4, respectively. The four plots in Figure 6 illustrate our empirical observations, which we now explain in detail.

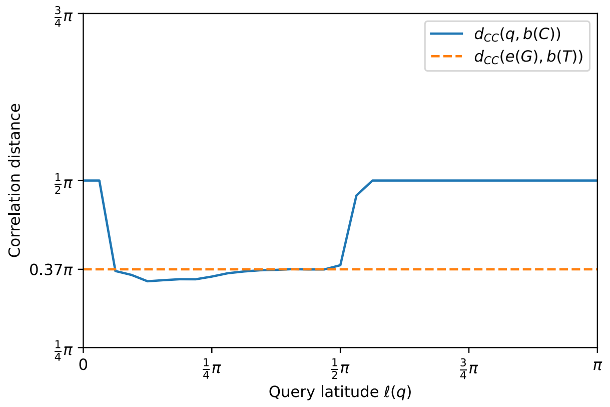
5. See `https://github.com/MartijnGosgens/hyperspherical_community_detection`.



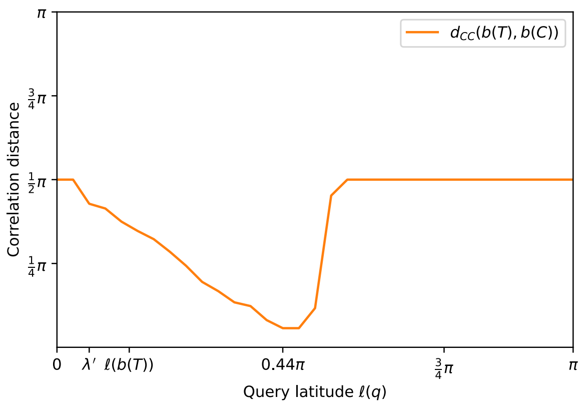
(a) The latitudes of the candidate and ground truth clusterings.



(b) Angular distances from the query vector to the ground truth and candidate clusterings.



(c) Correlation distances from the query vector to the ground truth and candidate clusterings.



(d) The performances of the query vectors. λ' is the query latitude that minimizes $d_a(\boldsymbol{q}, \boldsymbol{b}(T))$.

Figure 6: Plot of various quantities for a PPM consisting of 20 communities each of size 20, with mean intra-community degree 6 and inter-community degree 4. The query vectors are given by $\boldsymbol{q} = \mathcal{P}_\lambda(\boldsymbol{e}(G))$, where λ is varied between 0 and π, so that these query vectors lie on the ER meridian (see Equation 8). The candidate clusterings are obtained using the Louvain algorithm, i.e., $\boldsymbol{b}(C) = \mathcal{L}(\boldsymbol{q})$.

Quantity	Description
$\ell(\boldsymbol{b}(C))$	Latitude of the candidate clustering, measure of granularity.
$\ell(\boldsymbol{q})$	Latitude of the query vector. Related to the resolution parameter for modularity vectors.
$d_a(\boldsymbol{q}, \boldsymbol{b}(C))$	Angular distance between the query vector and candidate clustering, the quantity that is minimized by the Louvain algorithm.
$d_a(\boldsymbol{q}, \boldsymbol{b}(T))$	Angular distance between the query vector and the ground truth clustering.
$d_{\mathrm{CC}}(\boldsymbol{q}, \boldsymbol{b}(C))$	Correlation distance between the query vector and the candidate clustering.
$d_{\mathrm{CC}}(\boldsymbol{q}, \boldsymbol{b}(T))$	Correlation distance between the query vector and the ground truth clustering, measure of how informative $\boldsymbol{q}$ is for detecting T.
$d_{\mathrm{CC}}(\boldsymbol{b}(T), \boldsymbol{b}(C))$	Correlation distance between the ground truth and the candidate clusterings, the clustering performance measure used in this article.

Table 2: Description of quantities in terms of the query vector $\boldsymbol{q}$ and the corresponding Louvain candidate clustering $\boldsymbol{b}(C) = \mathcal{L}(\boldsymbol{q})$.

Candidate latitude. In Figure 6a, we show how the latitude of the candidate clustering changes with the latitude of the query vector. We see that $\ell(\boldsymbol{q}) = 0$ results in $\ell(\boldsymbol{b}(C)) = 0$. Indeed, since $\mathcal{P}_0(\boldsymbol{e}(G)) = -\mathbf{1}$ is a clustering vector, it follows that the Louvain algorithm maps this vector to itself since it is a projection onto the set of clustering vectors. Then the latitude of the candidate gradually increases until it roughly reaches the ground truth latitude (the dashed orange line) around the query latitude $\frac{1}{3}\pi$, plateaus for a bit and then sharply shoots up to π shortly after the candidate latitude $\frac{1}{2}\pi$. The behavior outside the plateau is exactly as one expects: in the part before the plateau, the Louvain algorithm only clusters together the highest-density subgraphs, while after the plateau, communities get merged. The sudden increase to π is explained by the fact that the corresponding candidate clustering consists of a single cluster, so that the corresponding clustering vector is $\mathbf{1}$, which has latitude π. Finally, note that before this sudden jump, the candidate latitude is significantly smaller than the query latitude, i.e., $\ell(\boldsymbol{b}(C)) < \ell(\boldsymbol{q})$. In general, we observe that in order to obtain a clustering of some specified latitude $\lambda_C < \frac{1}{2}\pi$, one needs to use a query latitude that is at least as large as λ_C.

Angular distances w.r.t. query vector. Figure 6b compares the angular distances between query vector and ground truth clustering, and between query vector and candidate clustering. The angular distance $d_a(\boldsymbol{q}, \boldsymbol{b}(T)))$ as a function of the query latitude $\ell(\boldsymbol{q})$ is given by

$$\cos d_a(\boldsymbol{q}, \boldsymbol{b}(T)) = \cos \ell(\boldsymbol{b}(T)) \cos \ell(\boldsymbol{q}) + \sin \ell(\boldsymbol{b}(T)) \sin \ell(\boldsymbol{q}) \cos d_{\mathrm{CC}}(\boldsymbol{e}(G), \boldsymbol{b}(T)),$$

as proven in Lemma 9 in Appendix A. From this expression, we see that when the query latitude equals 0, i.e., for $\boldsymbol{q} = -\mathbf{1}$, we have $d_a(\boldsymbol{q}, \boldsymbol{b}(T)) = \ell(\boldsymbol{b}(T))$. The angular distance to the ground truth then slightly decreases before it steadily increases to $\pi - \ell(\boldsymbol{b}(T))$ at query latitude π. The angular distance to the candidate clustering, on the other hand, starts at 0, exactly as in Figure 6a, and increases almost linearly to roughly $\frac{1}{2}\pi$, before it decreases linearly to 0. The linear decrease corresponds to the segment where $\ell(\boldsymbol{b}(C)) = \pi$, so that $d_a(\boldsymbol{q}, \boldsymbol{b}(C)) = \pi - \ell(\boldsymbol{q})$.

We observe in Figure 6b that the following two upper bounds for $d_a(\boldsymbol{q}, \boldsymbol{b}(C))$ hold in all cases: firstly, the upper bound $d_a(\boldsymbol{q}, \boldsymbol{b}(C)) \leq \ell(\boldsymbol{q})$ derived in (13) is confirmed. Secondly, the bound $d_a(\boldsymbol{q}, \boldsymbol{b}(C)) \leq d_a(\boldsymbol{q}, \boldsymbol{b}(T))$ holds in all cases. That is, the candidate clustering is always at least as close to the query vector as the ground truth clustering. Since Theorem 2 proves that minimizing the angular distance to the present query vector is equivalent to maximizing modularity for the ER null model, this means that the modularity of the candidate clustering is at least as high as that of the ground-truth clustering. This behavior has previously been observed in various real-world and synthetic networks (Prokhorenkova and Tikhonov, 2019), and may be explained by the fact that the modularity landscape is *glassy* (Good et al., 2010). That is, there are many clusterings with locally optimal modularity values very close to the global optimum. The greedy optimization that the Louvain algorithm utilizes seems to be quite successful in reaching one of those local optima, while the ground truth clustering is not guaranteed to correspond to a local modularity optimum at all. This phenomenon seems to hold even when minimizing the angular distance to other query vectors.

For query latitudes in the interval $[\frac{1}{4}\pi, \frac{1}{2}\pi]$, the inequality $d_a(\boldsymbol{q}, \boldsymbol{b}(C)) \leq d_a(\boldsymbol{q}, \boldsymbol{b}(T))$ seems to hold with equality, which suggests that there might be a corresponding lower bound. While we were unable to formally derive such a lower bound, we do observe that for many other networks and query mappings, there is a similar interval where $d_a(\boldsymbol{q}, \boldsymbol{b}(T)) \approx d_a(\boldsymbol{q}, \boldsymbol{b}(C))$ holds quite precisely. However, the endpoints of this interval do depend on the particular network and query mapping.

Correlation distances w.r.t. query vector. When comparing the correlation distances from the query vector to the ground truth and candidate clusterings in Figure 6c, we observe that $d_{\mathrm{CC}}(\boldsymbol{q}, \boldsymbol{b}(C)) \approx d_{\mathrm{CC}}(\boldsymbol{q}, \boldsymbol{b}(T))$ for a large range of query latitudes. This is interesting since the candidate clusterings in this range do have significantly different latitudes, as shown in Figure 6a. After that, the correlation distance jumps to $\frac{1}{2}\pi$ and stays constant. This is because the candidate clustering vector then corresponds to $\mathbf{1}$, which is a constant vector so that it has correlation distance $\frac{1}{2}\pi$ with any nonconstant vector.

Correlation distance between clusterings. Figure 6d shows the correlation distance between the ground truth and candidate clusterings. We use this measure to quantify the performance of the detection algorithm. We see that when the query latitude is zero, the correlation distance equals $\frac{1}{2}\pi$, indicating that the candidate clustering is uncorrelated to the ground truth. For higher query latitudes, this correlation distance decreases to a minimum at query latitude 0.44π, after which it increases back to $\frac{1}{2}\pi$ again. The best performance is thus obtained around a query latitude of 0.44π for this network.

If, in advance, one would guess what query latitude leads to the best performance, then the following two guesses may come to mind: on the one hand, one may think that setting the query latitude equal to the ground truth latitude $\ell(\boldsymbol{b}(T))$ would result in a candidate clustering of a similar latitude. On the other hand, the fact that the Louvain algorithm finds a clustering latitude close to the query vector suggests that we should place the query vector at minimal angular distance to $\boldsymbol{b}(T)$. Lemma 9 in Appendix A proves that the latitude λ' of this projection is given by $\tan \lambda' = \cos d_{\mathrm{CC}}(\boldsymbol{e}(G), \boldsymbol{b}(T)) \tan \ell(\boldsymbol{b}(T))$. It turns out that both of these options perform poorly in general, as shown in Figure 6d. Instead, the best performance seems to be achieved around the query latitude for which the candidate latitude intersects the ground truth latitude in Figure 6a. In Figure 6a, we also saw that the candidate latitude is generally significantly smaller than the query latitude, so that one indeed needs query latitudes much larger than $\ell(\boldsymbol{b}(T))$ in order to obtain a candidate clustering with $\ell(\boldsymbol{b}(C)) = \ell(\boldsymbol{b}(T))$. This tells us that both these initial guesses are wrong, and that instead we need to set the query latitude to some value larger than $\ell(\boldsymbol{b}(T))$ for the best performance.

We generally observe that the difference between the best-performing query latitude and the ground truth latitude is large whenever the correlation distance $d_{\mathrm{CC}}(\boldsymbol{q}, \boldsymbol{b}(T))$ between the query vector and the ground truth clustering is large. Indeed, when $d_{\mathrm{CC}}(\boldsymbol{q}, \boldsymbol{b}(T)) = 0$ and $\ell(\boldsymbol{q}) = \ell(\boldsymbol{b}(T))$, it also holds that $\boldsymbol{q} = \boldsymbol{b}(T)$, so that $\mathcal{L}(\boldsymbol{q}) = \boldsymbol{b}(T)$ follows from the fact that the Louvain algorithm is a projection to the set of clustering vectors. Of course, in practical applications, $d_{\mathrm{CC}}(\boldsymbol{q}, \boldsymbol{b}(T))$ and $\ell(\boldsymbol{b}(T))$ are unknown, making it difficult to know in advance which query latitude performs best. Finding this optimal query latitude in practical settings is beyond the scope of this paper.

6.2 Experiments on Real-World Networks

In this section, we compare the projection method for different query mappings on real-world networks. We consider both projection methods that fall inside the class of modularity-based methods and methods that fall outside this class. We consider four sets of query mappings, each corresponding to a straight line on the hypersphere that is parametrized by the query latitude. More specifically, we consider ER and CM modularity for various resolution parameters, the wedge-based query mappings from Section 5.2 for various latitudes, and the meridian corresponding to the CM-modularity vector for resolution parameter $\gamma = 1$, which we refer to as the *CM meridian.* Note that of these four lines on the hypersphere, CM modularity is the only one that does not correspond to a meridian.

For these experiments, we consider 6 real-world networks from Prokhorenkova and Tikhonov (2019) that have known ground truth communities. These networks are: 1) Zachary's well-known *karate club* network (Zachary, 1977); 2) Lusseau's network of bottlenose *dolphins* (Lusseau and Newman, 2004); 3) a network of *political books* grouped by political affiliation (Newman, 2006b); 4) a network of college *football* teams grouped by 'conference' (Girvan and Newman, 2002); 5) the *EU-core* network of European researchers linked by email traffic and grouped by department; and 6) a network of *political blogs* concerning the US presidential election of 2004, grouped by party affiliation (Adamic and Glance, 2005). An overview of these networks is given in Table 3.

The repository of Prokhorenkova and Tikhonov (2019) also contains two other networks: a network of Internet systems that are grouped by a geometric clustering algorithm, and a citation network that is grouped by a text-clustering algorithm. We chose not to include these two networks in the experiments since their corresponding 'ground truth' clusterings were obtained by different clustering algorithms, so that they may best be considered candidate clusterings rather than ground truths. Furthermore, these networks are significantly larger than the ones considered here ($n > 20,000$), which results in long running times for our modification[6] of the Louvain algorithm, especially when using the wedge-based query vector, as explained in Section 5.2.

The optimal query latitude. The results of the experiments can be found in Figure 7. We use the correlation distance between the ground truth clustering and the candidate clustering to measure the performance of the community detection method, where lower

6. We implemented our algorithm in Python, which is significantly slower than C++ for such tasks.

values indicate better performance. By comparing the locations of the minima for different type of query vectors (e.g., the wedges and ER meridian in Figure 7d), we see that the optimal query latitude is dependent on the network and query type. Furthermore, while the best-performing query vectors are often located roughly around the equator, there are many networks where the best latitude is quite far from the equator (e.g., Figures 7a and 7d). In Section 6.1, we have discussed that the optimal query latitude is generally larger than the ground truth latitude $\ell(\boldsymbol{b}(T))$. In Figure 7, we have marked the ground truth latitudes on the horizontal axes. It can be seen that the minima are indeed located at larger query latitudes, with the exceptions of the Football and Political blogs networks, where some minima seem to coincide with the ground truth latitude. Note that the exact location of these minima can only be determined when the ground truth is known, which is generally not the case in practice. Finding the best-performing query latitude without knowledge of the ground truth is beyond the scope of this article. In the remainder of this section, we compare the four methods by comparing their minima.

CM meridian versus modularity. Recall that CM modularity corresponds to a standard community detection method, while CM meridian is one of the possible alternative community detection methods that emerge from our generalization of modularity-based methods. Interestingly, CM meridian perfectly recovers the ground truth for Karate. In terms of Figure 3, this means that this meridian intersects the region of perfectly-performing query vectors. For the other networks, we see that the performances of CM meridian and CM modularity are comparable. The only exception is the EU-core, where CM modularity clearly outperforms CM meridian.

Correlation distance between the ground truth and query vectors. The correlation distance between the query vector and ground truth clustering vector seems to carry some information about which null model performs best: for all networks where CM modularity or meridian outperforms ER modularity, we see that $d_{\mathrm{CC}}(\boldsymbol{q}_M^{\mathrm{CM}}(G), \boldsymbol{b}(T)) < d_{\mathrm{CC}}(\boldsymbol{q}_M^{\mathrm{ER}}(G), \boldsymbol{b}(T))$ holds in Table 3. This suggests to use the correlation distance between the modularity vector and the ground truth as criteria to choose a suitable null model. However, a smaller correlation distance between the query vector and ground truth does not guarantee better performance. For example, $\boldsymbol{w}(G)$ is closer to the ground truth for all networks except EU-core (see Table 3), while the Football network is the only network for which the wedge vector actually outperforms the other vectors.

Performance of the wedge vector. As discussed in Section 5.2, we expected the wedge vector to perform well whenever the latitude of the ground truth vector is significantly larger than the latitude of the edge vector. We do not observe this to hold in the considered real-world networks: In Table 3, we see that for the Political blogs network, we have $\ell(\boldsymbol{b}(T)) = 0.500\pi > 0.096\pi = \ell(\boldsymbol{e}(G))$ so that we would expect the wedge vector to perform well, while it performs poorly compared to the other methods. Moreover, the only network for which the wedge vector visibly outperforms the other methods is the Football network for which $\ell(\boldsymbol{e}(G)) > \ell(\boldsymbol{b}(T))$. We expect that this difference in performance between the PPM of Section 5.2 and the real-world networks can be explained by degree-inhomogeneity: a pair of high-degree vertices are likely to have many common neighbors, regardless of whether they belong to the same community. This is similar to how ER modularity is known to cluster together high-degree vertices. This suggests 'correcting' for this degree-inhomogeneity in

Network	n	m_G	$\|T\|$	$\ell(\boldsymbol{b}(T))$	$\ell(\boldsymbol{e}(G))$	$d_{\mathrm{CC}}(\boldsymbol{q}_M^{\mathrm{ER}}(G), \boldsymbol{b}(T))$	$d_{\mathrm{CC}}(\boldsymbol{q}_M^{\mathrm{CM}}(G), \boldsymbol{b}(T))$	$d_{\mathrm{CC}}(\boldsymbol{w}(G), \boldsymbol{b}(T))$
Karate club	34	78	2	0.491π	0.243π	0.400π	0.388π	0.342π
Dolphins	62	159	2	0.536π	0.187π	0.420π	0.422π	0.364π
Political books	105	441	3	0.433π	0.183π	0.413π	0.414π	0.344π
Football	115	613	12	0.181π	0.198π	0.248π	0.248π	0.186π
EU-core	1005	16706	42	0.139π	0.116π	0.411π	0.403π	0.444π
Political blogs	1224	16718	2	0.500π	0.096π	0.461π	0.458π	0.424π

Table 3: Overview of the considered real-world networks. For each network, we show the number of vertices n, the number of edges m_G and the number of ground truth communities $|T|$. We also show the following angles: $\ell(\boldsymbol{b}(T))$, a measure of the granularity of the ground truth clustering; $\ell(\boldsymbol{e}(G))$, a measure of the edge-density of the network; and $d_{\mathrm{CC}}(\boldsymbol{q}, \boldsymbol{b}(T))$, the correlation distance between the ground truth clustering and the query vector, for $\boldsymbol{q} \in \{\boldsymbol{q}_M^{\mathrm{ER}}(G), \boldsymbol{q}_M^{\mathrm{CM}}(G), \boldsymbol{w}(G)\}$, where the modularity vectors use the default resolution parameter value $\gamma = 1$.

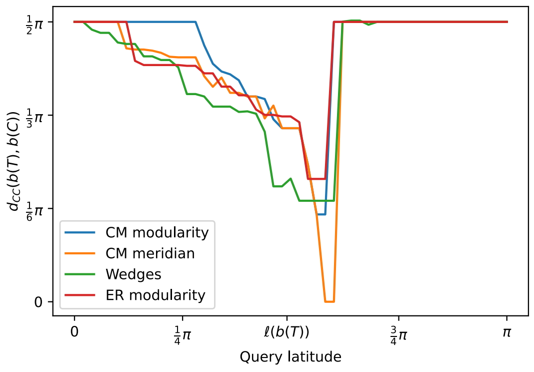
a similar way that CM modularity does: by subtracting a multiple of $d_i^{(G)} d_j^{(G)}$ from every entry.

In summary, these experiments illustrate how the developed geometry helps us to interpret results of community detection methods. In addition, we have demonstrated that the class of projection methods contains several new community detection methods based on CM meridian and the wedge vector as query mappings, which may outperform the subclass of modularity-based methods on real-world networks. Furthermore, we see that the correlation distance between the query vector and ground truth clustering can help in predicting which projection method will perform best.

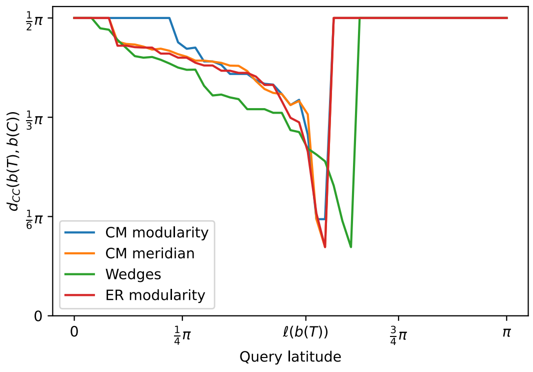
7. Discussion

In this work, we described a hyperspherical geometry on clusterings and showed how this geometry is related to validation measures such as the correlation distance. We then extended this geometry to include vectors that do not necessarily correspond to clusterings and proved that modularity maximization is equivalent to minimizing the distance to some modularity vector over the set of clustering vectors. In Section 4, we discussed how this newfound geometry sheds new light on modularity-based community detection methods: it allows us to view the popular Louvain algorithm as a method that projects a *query vector* onto the set of clustering vectors. In addition, this led to a geometric interpretation of the resolution limit. This geometry also suggests a generalization of modularity-based community detection methods, leading to the class of *projection methods* that detect communities by first mapping the graph to a point on the hypersphere, and then projecting this point to the set of clusterings. In Section 5, we introduced several projection methods that do not correspond to modularity-based methods. Finally, Section 6 applied our interpretation to real-world networks and demonstrated how our novel projection methods outperform existing modularity-based methods on several networks.

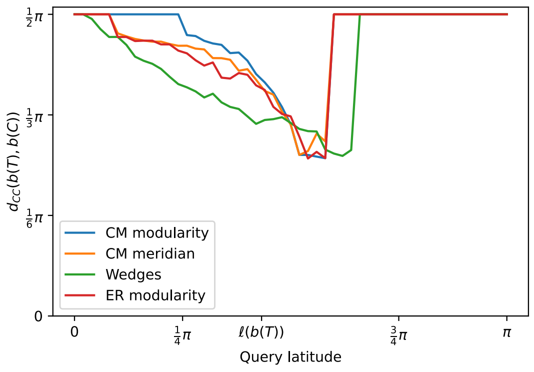
This work opens up many avenues for future research. In the remainder of this section, we discuss the ones that are most promising in our opinion.



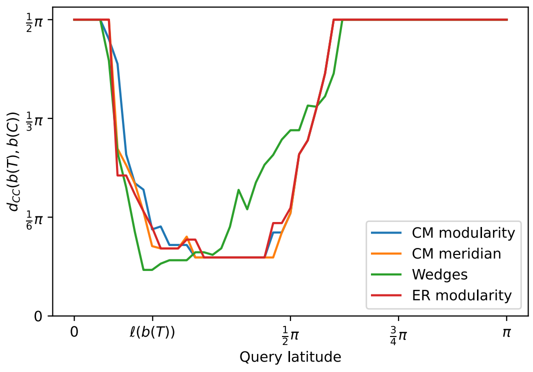
(a) Performance for Karate



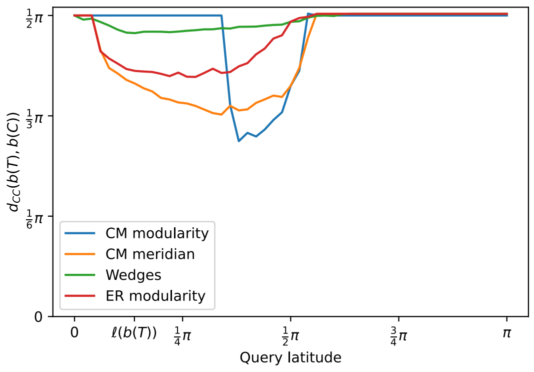
(b) Performance for Dolphins



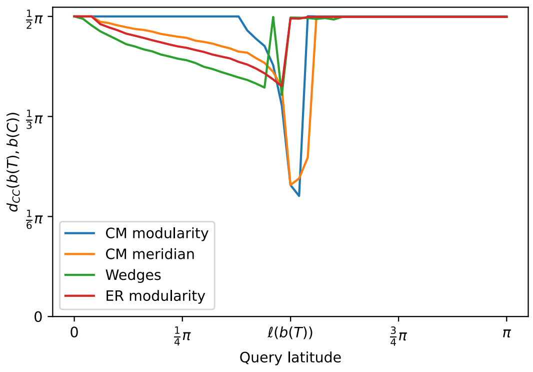
(c) Performance for Political books



(d) Performance for Football



(e) Performance for EU-core



(f) Performance for Political blogs

Figure 7: Results of the various projection methods on the real-world networks summarized in Table 3. The ground truth latitude is marked on the horizontal axis.

Finding a suitable query vector. There are infinitely many possible query mappings and finding the most suitable one is a daunting task. The described geometry allows us to split the selection of a query vector into two subtasks: first finding a suitable meridian and then finding a suitable latitude on that meridian. For the first problem, a reasonable approach could be to search for the meridian that minimizes the correlation distance to the ground truth clustering. Since the correlation distance is the angle between the meridians (recall Theorem 1), it does not depend on the query latitude that we choose in the second step. However, we have shown in Section 6.2 that this criterion does not always result in better performances. The problem of finding the optimal latitude on a given meridian seems simpler, as it is one-dimensional. We have observed in Section 6 that the optimal query latitude is generally larger than the ground truth clustering latitude, and that the difference between these two is larger whenever the query vector and ground truth clustering are poorly correlated. However, in Section 6.2 we have also demonstrated that the optimal query latitude depends on the particular query mapping, the granularity of the ground truth clustering and properties of the network at hand. This makes it difficult to prescribe a general formula for the query latitude that performs well across all these possible variations. A further analysis of optimal latitudes is an interesting direction for future research.

Projections in other geometries. We have proved that maximizing modularity is equivalent to minimizing the angular distance to a query vector. This begs the question whether minimizing other distances would also give good community detection methods. In particular, it would be interesting to investigate minimizing the correlation distance instead of the angular distance, as this distance would allow one to use the triangle inequality to bound the correlation distance between candidate and ground truth clusterings by their correlation distances to the query vector. Furthermore, since the correlation distance is a distance between meridians (by Theorem 1), such methods would be invariant to the choice of the query latitude.

Improved modularity optimization algorithms. Finally, as we have proven that the Louvain algorithm is an approximate nearest-neighbor algorithm, it would be interesting to see whether existing approximate nearest-neighbor algorithms outperform the Louvain algorithm in terms of the obtained modularity value or in terms of running time. At any rate, it may be worthwile to investigate whether the geometry may be utilized to improve upon Louvain or other modularity-optimizing algorithms.

Acknowledgments

This work is supported in part by the Netherlands Organisation for Scientific Research (NWO) through the Gravitation NETWORKS grant no. 024.002.003.

Appendix A. Additional Proofs

Lemma 7 *The latitude of the modularity vector for the Erdős-Rényi null model is given by*

$$\tan \ell(\boldsymbol{q}_M^{ER}(G;\gamma)) = \frac{\sqrt{\frac{N-m_G}{m_G}}}{\gamma - 1}.$$

Proof Note that $\boldsymbol{p}^{\mathrm{ER}}(G) = \frac{m_G}{N}\mathbf{1}$, so that $\|\boldsymbol{p}^{\mathrm{ER}}(G)\|^2 = \frac{m_G^2}{N}$ and

$$\langle \boldsymbol{p}^{\mathrm{ER}}(G), \boldsymbol{e}(G)\rangle = 2\frac{m_G^2}{N} - m_G.$$

We substitute these into (14) and rewrite the result to

$$\begin{aligned}\cos\ell(\boldsymbol{q}_M^{\mathrm{ER}}(G;\gamma)) &= \frac{(\gamma-1)m_G}{\sqrt{N}\cdot\sqrt{(1-\gamma)m_G + \gamma^2\frac{m_G^2}{N} - \gamma(2\frac{m_G^2}{N} - m_G)}} \\ &= \frac{(\gamma-1)\frac{m_G}{N}}{\sqrt{\left((\gamma-1)\frac{m_G}{N}\right)^2 + \frac{m_G(N-m_G)}{N^2}}}.\end{aligned}$$

Therefore, the tangent is given by

$$\tan\ell(\boldsymbol{q}_M^{\mathrm{ER}}(G;\gamma)) = \frac{\sqrt{1-\cos^2\ell(\boldsymbol{q}_M^{\mathrm{ER}}(G;\gamma))}}{\cos\ell(\boldsymbol{q}_M^{\mathrm{ER}}(G;\gamma))} = \frac{\sqrt{\frac{m_G(N-m_G)}{N^2}}}{(\gamma-1)\frac{m_G}{N}} = \frac{\sqrt{\frac{N-m_G}{m_G}}}{\gamma-1},$$

as required. ■

Lemma 8 *The Louvain algorithm is a projection. That is, $\mathcal{L}(\mathcal{L}(\boldsymbol{q})) = \mathcal{L}(\boldsymbol{q})$ holds for any query vector $\boldsymbol{q} \in \mathbb{R}^N$.*

Proof We equivalently prove that for any clustering C, the Louvain algorithm maps the query vector $\boldsymbol{q} = \boldsymbol{b}(C)$ to itself, i.e., $\mathcal{L}(\boldsymbol{b}(C)) = \boldsymbol{b}(C)$. Recall that the Louvain algorithm is initialized at the fine pole, i.e., the clustering consisting of n singleton clusters. Then, it iterates through all vertices and relabels each vertex greedily. That is, a vertex is assigned to the cluster that results in the largest decrease in the angular distance to the query vector, or equivalently, the largest increase of $\langle \boldsymbol{b}(C'), \boldsymbol{b}(C)\rangle$, where C' is the new candidate clustering after relabeling.

Let us consider one cluster $c \in C$. The first time that the iteration of the Louvain algorithm encounters a vertex $i \in c$, all of its cluster-members are assigned to singleton clusters, so that i is relabeled to the cluster of one of its cluster-members $j \in c\setminus\{i\}$ arbitrarily. Then, the next time a vertex $k \in c \setminus \{i,j\}$ is encountered in the iteration, the greedy choice is to relabel k to the cluster $\{i,j\}$, as it results in a larger increase than relabeling k to any of the singleton clusters of $c \setminus \{i,j,k\}$. Similarly, all other vertices of c are relabeled to this cluster so that the resulting Louvain candidate contains the cluster c. In the same way, all other clusters $c' \in C$ are obtained so that indeed $\mathcal{L}(\boldsymbol{b}(C)) = \boldsymbol{b}(C)$. ■

Lemma 9 *For a fixed vector $\boldsymbol{y} \in \mathbb{R}^N$ with latitude $\lambda_y = \ell(\boldsymbol{y}) \in (0,\pi)$, another vector $\boldsymbol{x} \in \mathbb{R}^N$, at correlation distance $\theta = d_{\mathrm{CC}}(\boldsymbol{x},\boldsymbol{y}) \in (0,\frac{1}{2}\pi)$ and a latitude $\lambda \in (0,\pi)$, the angular distance between the parallel projection $\mathcal{P}_\lambda(\boldsymbol{x})$ and $\boldsymbol{y}$ is given by*

$$\cos d_a(\mathcal{P}_\lambda(\boldsymbol{x}), \boldsymbol{y}) = \cos\lambda\cos\lambda_y + \sin\lambda\sin\lambda_y\cos\theta. \tag{15}$$

In particular, the λ that minimizes the angular distance to $\boldsymbol{y}$ is given by

$$\tan\lambda = \cos\theta\tan\lambda_y. \tag{16}$$

Proof By the definition of the correlation distance as given in (9),

$$\cos\theta = \frac{\cos d_a(\mathcal{P}_\lambda(\boldsymbol{x}), \boldsymbol{y}) - \cos\lambda\cos\lambda_y}{\sin\lambda\sin\lambda_y}.$$

This can be rewritten to obtain (15). Taking the derivative of (15) with respect to λ, we get

$$\frac{d}{d\lambda} d_a(\mathcal{P}_\lambda(\boldsymbol{x}), \boldsymbol{y}) = -\sin\lambda\cos\lambda_y + \cos\lambda\sin\lambda_y\cos\theta.$$

The unique zero of this expression is given by (16), which concludes the proof. ■

Appendix B. Heatmap of Dolphins Network

Figure 8 shows a similar experiment as in Section 5.1 performed on Lusseau's network of bottlenose dolphins (Lusseau and Newman, 2004). Similarly to the karate network, this experiment shows that there is a small region of query vectors for which perfect recovery is achieved. This time, this region is found on meridians that correspond to CM modularity for a negative resolution parameter: one of these query vectors is given by $\boldsymbol{q} = \mathcal{P}_{0.58\pi}(\boldsymbol{q}_M^{\mathrm{CM}}(G; -0.2))$, i.e., CM modularity with resolution $\gamma = -0.2$ projected to the latitude 0.58π. Note that if we were to try and interpret this query vector in terms of a null model, the expected number of edges between two vertices i, j would be of the form $c_1 - c_2 d_i^{(G)} d_j(G)$ for $c_1, c_2 > 0$. For some vertex-pairs, this value may be negative, so that it cannot be interpreted as an expected number of edges. The reason that this query vector performs well on this network seems to be that all high-degree vertices are part of the same community. Therefore, vertex-pairs for which $d_i^{(G)} d_j^{(G)}$ is high are more likely to be community members.

References

Lada A Adamic and Natalie Glance. The political blogosphere and the 2004 US election: divided they blog. In *Proceedings of the 3rd international workshop on Link discovery*, pages 36–43, 2005.

Gaurav Agarwal and David Kempe. Modularity-maximizing graph communities via mathematical programming. *The European Physical Journal B*, 66(3):409–418, 2008.

Ahmed N Albatineh, Magdalena Niewiadomska-Bugaj, and Daniel Mihalko. On similarity indices and correction for chance agreement. *Journal of Classification*, 23(2):301–313, 2006.

Alex Arenas, Alberto Fernandez, and Sergio Gomez. Analysis of the structure of complex networks at different resolution levels. *New Journal of Physics*, 10(5):053039, 2008.

Albert-László Barabási. Network science. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1987):20120375, 2013.

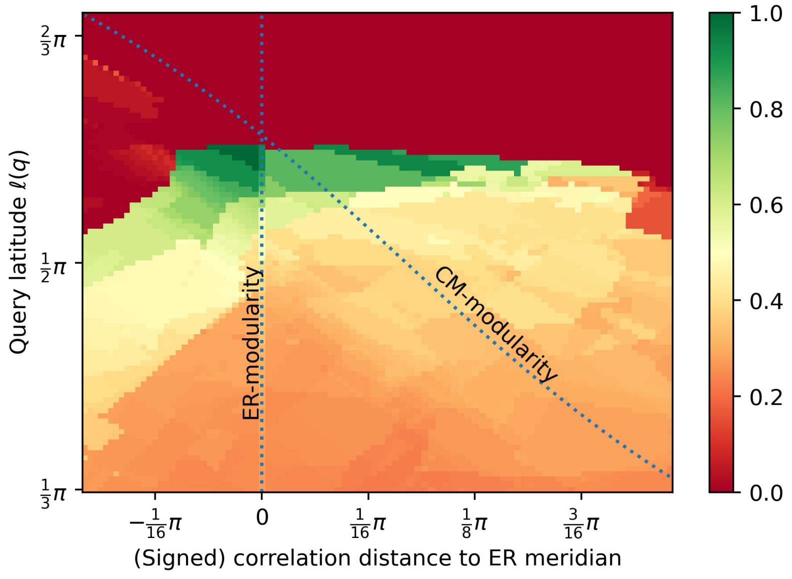


Figure 8: Heatmap of the correlation between ground truth and candidate clusterings for the Dolphins network (Lusseau and Newman, 2004). We take query vectors from the meridians that $\left(\boldsymbol{q}_M^{\mathrm{CM}}(G;\gamma)\right)_{\gamma\in[-1.5,2]}$ runs through and vary the query latitude between $\frac{1}{3}\pi$ and $\frac{2}{3}\pi$. The horizontal coordinates are given by $\mathrm{sgn}(\gamma)\cdot d_{\mathrm{CC}}(\boldsymbol{q},\boldsymbol{e}(G))$, i.e., the (signed) correlation distance to the ER meridian.

Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Gorke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. On modularity clustering. *IEEE Transactions on Knowledge and Data Engineering*, 20(2):172–188, 2007.

Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical Review E*, 70(6):066111, 2004.

Ronald R Coifman, Stephane Lafon, Ann B Lee, Mauro Maggioni, Boaz Nadler, Frederick Warner, and Steven W Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *Proceedings of the National Academy of Sciences*, 102(21):7426–7431, 2005.

Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. A survey on network embedding. *IEEE Transactions on Knowledge and Data Engineering*, 31(5):833–852, 2018.

J-C Delvenne, Sophia N Yaliraki, and Mauricio Barahona. Stability of graph communities across time scales. *Proceedings of the National Academy of Sciences*, 107(29):12755–12760, 2010.

Joseph D Donnay. *Spherical trigonometry*. Read Books Ltd, 2011.

Michaël Fanuel, Antoine Aspeel, Jean-Charles Delvenne, and Johan AK Suykens. Positive semi-definite embedding for dimensionality reduction and out-of-sample extensions. *SIAM Journal on Mathematics of Data Science*, 4(1):153–178, 2022.

Maurizio Filippone, Francesco Camastra, Francesco Masulli, and Stefano Rovetta. A survey of kernel and spectral methods for clustering. *Pattern Recognition*, 41(1):176–190, 2008.

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

Santo Fortunato and Marc Barthélemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36–41, 2007. ISSN 0027-8424.

Santo Fortunato and Darko Hric. Community detection in networks: A user guide. *Physics Reports*, 659:1–44, 2016.

Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

Benjamin H Good, Yves-Alexandre De Montjoye, and Aaron Clauset. Performance of modularity maximization in practical contexts. *Physical Review E*, 81(4):046106, 2010.

Martijn M Gösgens, Alexey Tikhonov, and Liudmila Prokhorenkova. Systematic analysis of cluster similarity indices: How to validate validation measures. In *International Conference on Machine Learning*, pages 3799–3808, 2021.

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge Discovery and Data mining*, pages 855–864, 2016.

Albert Gu, Frederic Sala, Beliz Gunel, and Christopher Ré. Learning mixed-curvature representations in product spaces. In *International Conference on Learning Representations*, 2018.

Roger Guimera and Luís A Nunes Amaral. Functional cartography of complex metabolic networks. *Nature*, 433(7028):895–900, 2005.

Lawrence Hubert. Nominal scale response agreement as a generalized correlation. *British Journal of Mathematical and Statistical Psychology*, 30(1):98–103, 1977.

Paul R Hunter and Michael A Gaston. Numerical index of the discriminatory ability of typing systems: an application of simpson's index of diversity. *Journal of Clinical Microbiology*, 26(11):2465–2466, 1988.

Paul Jaccard. The distribution of the flora in the alpine zone. 1. *New Phytologist*, 11(2): 37–50, 1912.

Anil K Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8): 651–666, 2010.

Jussi M Kumpula, Jari Saramäki, Kimmo Kaski, and János Kertész. Limited resolution in complex network community detection with potts model approach. *The European Physical Journal B*, 56(1):41–45, 2007.

Stephane Lafon and Ann B Lee. Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1393–1403, 2006.

Andrea Lancichinetti and Santo Fortunato. Limits of modularity maximization in community detection. *Physical Review E*, 84(6):066122, 2011.

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

Juyong Lee, Steven P Gross, and Jooyoung Lee. Modularity optimization by conformational space annealing. *Physical Review E*, 85(5):056702, 2012.

Yang Lei, James C. Bezdek, Simone Romano, Nguyen Xuan Vinh, Jeffrey Chan, and James Bailey. Ground truth bias in external cluster validity indices. *Pattern Recognition*, 65:58 – 70, 2017. ISSN 0031-3203.

Zijing Liu and Mauricio Barahona. Geometric multiscale community detection: Markov stability and vector partitioning. *Journal of Complex Networks*, 6(2):157–172, 2018.

David Lusseau and Mark EJ Newman. Identifying the role that animals play in their social networks. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 271 (suppl_6):S477–S481, 2004.

Mark EJ Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74(3):036104, 2006a.

Mark EJ Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006b.

Mark EJ Newman. Equivalence between modularity optimization and maximum likelihood methods for community detection. *Physical Review E*, 94(5):052315, 2016.

Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2):026113, 2004.

Liudmila Prokhorenkova. Using synthetic networks for parameter tuning in community detection. In *International Workshop on Algorithms and Models for the Web-Graph*, pages 1–15. Springer, 2019.

Liudmila Prokhorenkova and Alexey Tikhonov. Community detection through likelihood optimization: in search of a sound model. In *The World Wide Web Conference*, pages 1498–1508, 2019.

William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

Jörg Reichardt and Stefan Bornholdt. Statistical mechanics of community detection. *Physical Review E*, 74(1):016110, 2006.

Simone Romano, Nguyen Xuan Vinh, James Bailey, and Karin Verspoor. Adjusting for chance clustering comparison measures. *The Journal of Machine Learning Research*, 17 (1):4635–4666, 2016.

Martin Rosvall, Jean-Charles Delvenne, Michael T Schaub, and Renaud Lambiotte. Different approaches to community detection. *Advances in Network Clustering and Blockmodeling*, pages 105–119, 2019.

Daniel López Sánchez, Jorge Revuelta, Fernando De la Prieta, Ana B Gil-González, and Cach Dang. Twitter user clustering based on their preferences and the louvain algorithm. *Trends in Practical Applications of Scalable Multi-Agent Systems, the PAAMS Collection*, pages 349–356, 2016.

Michael T Schaub, Jean-Charles Delvenne, Renaud Lambiotte, and Mauricio Barahona. Multiscale dynamical embeddings of complex networks. *Physical Review E*, 99(6):062308, 2019.

Christian L Staudt, Aleksejs Sazonovs, and Henning Meyerhenke. Networkit: A tool suite for large-scale complex network analysis. *Network Science*, 4(4):508–530, 2016.

Clara Stegehuis, Remco van der Hofstad, and Johan SH van Leeuwaarden. Power-law relations in random networks with communities. *Physical Review E*, 94(1):012302, 2016.

Isaac Todhunter. *Spherical trigonometry, for the use of colleges and schools: with numerous examples*. Macmillan, 1863.

Vincent A Traag and Jeroen Bruggeman. Community detection in networks with positive and negative links. *Physical Review E*, 80(3):036115, 2009.

Vincent A Traag, Paul van Dooren, and Yurii Nesterov. Narrow scope for resolution-limit-free community detection. *Physical Review E*, 84(1):016114, 2011.

Vincent A Traag, Ludo Waltman, and Nees Jan van Eck. From louvain to leiden: guaranteeing well-connected communities. *Scientific Reports*, 9(1):1–12, 2019.

Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *Proceedings of the 26th annual International Conference on Machine Learning*, pages 1073–1080, 2009.

Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *The Journal of Machine Learning Research*, 11:2837–2854, 2010.

Ivan Voitalov, Pim van der Hoorn, Remco van der Hofstad, and Dmitri Krioukov. Scale-free networks well done. *Physical Review Research*, 1(3):033034, 2019.

Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4): 395–416, 2007.

Ken Wakita and Toshiyuki Tsurumi. Finding community structure in mega-scale social networks. In *Proceedings of the 16th international conference on World Wide Web*, pages 1275–1276, 2007.

Rui Xu and Donald Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005.

Wayne W Zachary. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, 33(4):452–473, 1977.

Xiao Zhang and Mark EJ Newman. Multiway spectral community detection in networks. *Physical Review E*, 92(5):052808, 2015.