A Weighted Projection Framework for Hypergraph Diffusion: From Bipartite Walks to Fast Community Detection

PHAN Thi Ha Duong

ICRTM - Institute of Mathematics, Vietnam Academy of Science and Technology, Vietnam
phanhaduong@math.ac.vn

Abstract

In modern graph theory, hypergraphs effectively model complex group interactions. However, traditional unweighted projections from hypergraphs to ordinary graphs discard critical structural information. We introduce a theoretical framework of dual weighted projections induced by natural random walks, yielding dual weighted vertex and hyperedge graphs. We establish that both projections share identical non zero spectra, allowing spectral operations and random-walk metrics (such as trajectory distances and hitting times) to be computed in the smaller hyperedge space when hyperedges are far fewer than vertices. In addition, our weighted projections overcome the resolution limit of standard modularity, empowering community detection algorithms like Louvain and Leiden to isolate micro communities. Our approach also supports Euclidean graph embeddings and naturally generalizes to multilayer networks.

1 Introduction

Many complex systems are naturally represented using graph structures, in that the edges represent interactions between two objects. However, in several natural and artificial systems, interactions are not merely pairwise but frequently occur among groups of multiple entities simultaneously. This necessitates the use of hypergraphs, where a single hyperedge can connect an arbitrary number of vertices. Hypergraphs provide a powerful modeling tool across various domains, including biological protein complex networks [23, 25], information-entropy-based social network partitioning [31], as well as general machine learning tasks such as classification and clustering [32].

In the graph representation, numerous algorithms have been developed to measure vertex similarity and identify cohesive community structures [1, 5, 8, 12, 15, 19, 20, 24, 27, 28]. To leverage the vast array of existing graph-theoretic tools, a standard approach is to project the hypergraph onto a simple weighted graph. Traditional projection techniques define that there is a (unweighted) edge between two vertices if they are in at least one hyperedge (see, for example, [30]). This projection discards rich structural information regarding group exclusivity. To address this limitation, several recent studies have incorporated random walk dynamics to weight projected edges. For instance, Zhou et al. [32] and Hayashi et al. [14] investigated hypergraph Laplacians and spectral analysis, Chitra and Raphael [2] analyzed Markov chain properties such as mixing time and stationary distributions, Wang et al. [29] utilized random

walks to discover hidden properties in bipartite web graphs, and Eriksson et al. [11] analyzed higher order information flows for community detection.

In our work, we develop the idea of weighted projection by using natural random walks on hypergraphs, but go further through several core contributions:

- Two Projections on vertices (G_V) and on hyperedges ($G_{\mathcal{C}}$): we explicitly formulate weighted projection matrices for both the vertex graph $G_V = (V, E_V, w_V)$ and the hyperedge graph $G_{\mathcal{C}} = (\mathcal{C}, E_{\mathcal{C}}, w_{\mathcal{C}})$, which correspond to the random walk on the hypergraph and induce random walks on G_V and $G_{\mathcal{C}}$. Constructing $G_{\mathcal{C}}$ alongside G_V is particularly advantageous in real-world applications. Many network models are naturally encoded by their maximal covering cliques, and in numerous domain networks, the number of hyperedges is substantially smaller than the number of vertices ($\rfloor \ll n$), such as functional protein modules relative to genes or movie categories relative to viewers. In such settings, performing computations on $G_{\mathcal{C}}$ rather than G_V significantly reduces space and computational complexity.

- Explicit Embedding and Spectral Equivalence: we derive an explicit structural embedding into Euclidean space ($\mathbb{R}^n$ or $\mathbb{R}^r$), providing a natural dimensionality reduction scheme. Under our cardinality-normalized weighting scheme $w_V(u, v) = \sum_{c \ni u,v} \frac{1}{d_H(c)}$, the spectra of G_V and $G_{\mathcal{C}}$ are strictly identical (up to zero eigenvalues). Consequently, key properties and structural dynamics of the larger vertex graph G_V can be efficiently computed via the smaller hyperedge graph $G_{\mathcal{C}}$.

- Trajectory Based Structural Distances: we define a concrete structural distance metric between vertices based on multi step random walk transition probabilities, such as Walktrap dynamics and hitting times, note that Walktrap and hitting times are widely studied for community detection algorithms [22, 6]. These distances help establish a rigorous geometric foundation for spatial clustering algorithms (e.g., K-means or PCA algorithms).

- Modularity Optimization for Small Community Discovery: modularity is a commonly used measure for evaluating the quality of a clustering [17, 21], but it suffers from limitations; for instance, for small communities, the modularity contribution can be negligible, so algorithms based on modularity optimization such as Louvain [1] or Leiden [27] may fail to discover small communities. We evaluate community quality using weighted graph modularity. We demonstrate that our specific weighted projection enables modularity optimization to uncover small, highly localized community structures that conventional unweighted projections fail to detect.

The remainder of this paper is organized as follows. In Section 2, we establish the formal model for group-based networks and hypergraph dual projections. In Section 3, we analyze modularity in weighted graphs, demonstrating how our weighted projection resolves the size-imbalance limitation of unweighted modularity. Section 4 investigates random walk dynamics on group based structures, we derive trajectory based distances on G_V and prove that their computations can be significantly accelerated via the spectrum of the smaller hyperedge operators (in Section 5). In Section 6, we introduce our integrated community detection framework, detailing a multi level dimensionality reduction scheme, from vertex space down to the principal component space, for K means and related algorithms. Section 7 extends our framework beyond bipartite/hypergraph structures to tripartite and arbitrary multi layer architectures.

2 Dual weighted projections of hypergraphs

Many real world systems are organized around groups rather than pairwise interactions. Examples include author-paper networks, user-movie networks, students enrolled in courses, and individuals participating in social communities. In such systems, relationships are naturally established through the participation of several entities in common groups.

These structures can be modeled by a hypergraph $H = (V, \mathcal{C})$, where V is a finite set of vertices and $\mathcal{C}$ is a collection of subsets of V, called hyperedges or groups. A vertex $v \in V$ belongs to a group $c \in \mathcal{C}$ whenever $v \in c$. Denote by n the cardinality of V, and by $\mathbf{c}$ the cardinality of $\mathcal{C}$

A standard representation of a hypergraph is its incidence bipartite graph.

Definition 2.1 (Incidence bipartite graph) *Let $H = (V, \mathcal{C})$ be a hypergraph where $\mathcal{C} \subset 2^V$. The incidence bipartite graph associated with H is the bipartite graph $G_B = (V \cup \mathcal{C}, E_B)$, where the vertex set is the disjoint union of V and $\mathcal{C}$, and for $v \in V, c \in \mathcal{C}$: $(v, c) \in E_B \iff v \in c$.*

The degree of an element in H is defined as its degree in the corresponding bipartite graph, that means the number of incidence edges in the bipartite of this element. For $v \in V$, its degree is $d_H(v) = d_B(v) = |\{c \in \mathcal{C} : \ v \in c\}|$, i.e., the number of groups containing v. Similarly, for $c \in \mathcal{C}$, its degree is $d_H(c) = d_B(c) = |\{v \in V : \ v \in c\}|$, i.e., the number of vertices contained in c. Without ambiguity, we can write $d(v)$ and $d(c)$ instead of $d_H(v)$ and $d_H(c)$.

The bipartite representation preserves the complete incidence structure of the original hypergraph and provides a convenient framework for applying graph theoretical tools.

A naive projection connects two vertices whenever they share at least one group. However, this ignores important structural effects: large groups should contribute less, and vertices participating in many groups should distribute their influence across multiple interactions. On the other hand, in many realht!world systems, the fact that two entities belong to the same group does not necessarily imply that their relationship becomes stronger as the size of the group increases. For instance, in collaboration networks, two researchers coauthoring a paper with only a few authors are typically more closely connected than two researchers appearing in a paper with hundreds of authors. Similar situations arise in project teams, classrooms, or online communities, where membership in a large group often provides weaker evidence of a direct relationship between two specific individuals than membership in a small and tightly connected group. A comparable phenomenon is observed in several natural and information systems. In molecular biology, two proteins belonging to a large protein complex are not necessarily more strongly related than two proteins participating in a small functional module. In ecology, the co-occurrence of two species in a highly diverse habitat generally provides less information about their ecological similarity than their co-occurrence in a small and specialized environment. Likewise, in information retrieval, two documents sharing a very common keyword are considered less related than two documents sharing a rare and highly specific term.

These examples suggest viewing each hyperedge as carrying a finite amount of information or influence. This motivates a normalization with respect to the hyperedge size, whereby the contribution of each hyperedge is evenly distributed among its members. These observations motivate the following refined projection.

Definition 2.2 (Weighted projection onto V) *Let $H = (V, \mathcal{C})$ be a hypergraph. The weighted projection onto V is the weighted directed graph $G_V = (V, E_V, w_V)$, where $E_V = \big\{\{u, v\} \subseteq$*

$V : \exists c \in \mathcal{C},\ u, v \in c\}$, *and the weight is defined by* $w_V(u,v) = \sum_{c:\, u,v \in c} \frac{1}{d_H(c)}$, $w_V(u,u) = \sum_{c:\, u \in c} \frac{1}{d_H(c)}$, *and* $d_V(u) = \sum_{v \in V} w_V(u,v)$.

Note that in the symmetric directed graph G_V, *each edge* (u,v) *means a directed edge from* u *to* v *and a directed edge from* v *to* u *with the same weight* $w_V(u,v)$. *Moreover, the degree* $d_V(u)$ *means the outgoing degree of* u.

Note that in the directed graph G_V, each edge (u,v) means a directed edge from u to v and an directed edge from v to u with the same weight. Moreover, the degree $d_V(u)$ means te outgoing degree of u.

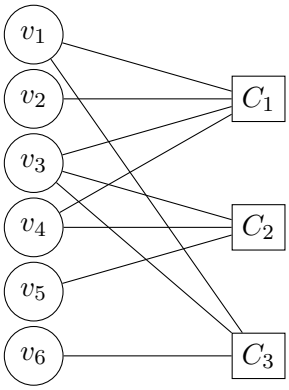
A natural question is whether a similar construction can be defined on the group side $\mathcal{C}$. In many applications, it is equally important to understand relationships between groups themselves. For instance, in an author-paper network, one may wish to quantify the similarity between two papers based on their coauthors. In a user-item system, this corresponds to measuring the similarity between items via their user base. This leads to a dual projection defined on $\mathcal{C}$.

Definition 2.3 (Weighted projection onto $\mathcal{C}$) *Let* $H = (V, \mathcal{C})$ *be a hypergraph. The weighted projection onto* $\mathcal{C}$ *is the weighted graph* $G_{\mathcal{C}} = (\mathcal{C}, E_{\mathcal{C}}, w_{\mathcal{C}})$, *where* $E_{\mathcal{C}} = \big\{\{c,d\} \subseteq \mathcal{C} : \exists v \in V,\ v \in c \cap d\big\}$, *and the weight is defined by* $w_{\mathcal{C}}(c,d) = \sum_{v \in c \cap d} \frac{1}{d(v)}$, $w_{\mathcal{C}}(c,c) = \sum_{v \in c} \frac{1}{d(v)}$ *and* $d_{\mathcal{C}} c = \sum_{d \in \mathcal{C}} w_{\mathcal{C}}(c,d)$.

Note that in the symmetric directed graph $G_{\mathcal{C}}$, *each edge* (c,d) *means a directed edge from* c *to* d *and a directed edge from* d *to* c *with the same weight* $w_{\mathcal{C}}(c,d)$. *Moreover, the degree* $d_{\mathcal{C}}(c)$ *means the outgoing degree of* c.

Equivalently, letting B be the incidence matrix of G_B, W_V, $W_{\mathcal{C}}$ be the weighted matrices and D_V, $D_{\mathcal{C}}$ be the diagonal degree matrices of G_V and $G_{\mathcal{C}}$, we can write: $W_V = B D_{\mathcal{C}}^{-1} B^\top, W_{\mathcal{C}} = B^\top D_V^{-1} B$. We now illustrate the construction through a concrete example.

Example 2.4 *Consider the hypergraph* $H = (V, \mathcal{C})$ *with* $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$, *and* $\mathcal{C} = \{C_1 = \{1,2,3,4\}, C_2 = \{3,4,5\}, C_3 = \{1,3,6\}\}$. *We compute degrees:* $|C_1| = 4, |C_2| = 3, |C_3| =$



$$B = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Figure 1: Incidence bipartite graph G_B and its incidence matrix B.

3, *and* $d(v_1) = 2,\ d(v_2) = 1,\ d(v_3) = 3,\ d(v_4) = 2,\ d(v_5) = 1,\ d(v_6) = 1$. *The weighted projection*

onto V and on $\mathcal{C}$ are given by

$$W_V = \begin{pmatrix} \frac{7}{12} & \frac{1}{4} & \frac{7}{12} & \frac{1}{4} & 0 & \frac{1}{3} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & 0 \\ \frac{7}{12} & \frac{1}{4} & \frac{11}{12} & \frac{7}{12} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{4} & \frac{1}{4} & \frac{7}{12} & \frac{7}{12} & \frac{1}{3} & 0 \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \\ \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} \end{pmatrix}, \qquad W_{\mathcal{C}} = \begin{pmatrix} \frac{7}{3} & \frac{5}{6} & \frac{5}{6} \\ \frac{5}{6} & \frac{11}{6} & \frac{1}{3} \\ \frac{5}{6} & \frac{1}{3} & \frac{11}{6} \end{pmatrix}.$$

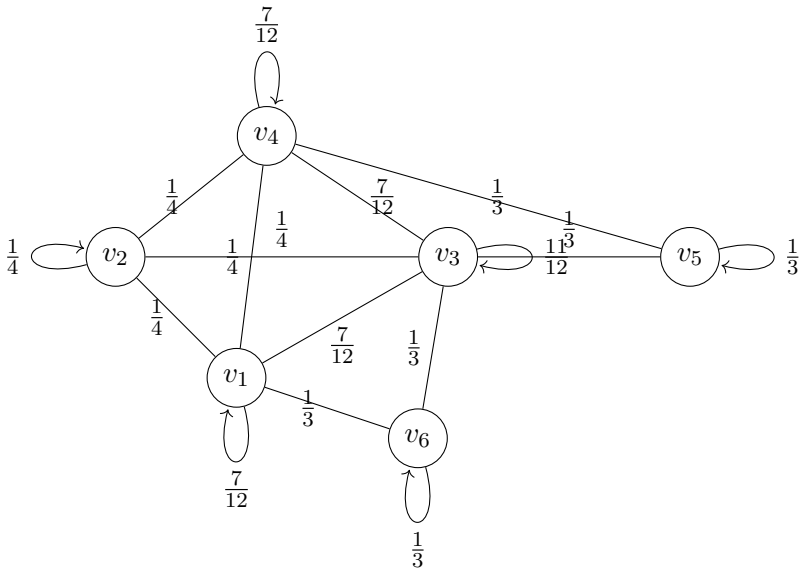


Figure 2: Weighted version of the projected graph G_V.

Remark 2.5 *For the above example, the weighted degrees of the vertices in G_V are $(2, 1, 3, 2, 1, 1)$, which coincide exactly with the degrees of the corresponding vertices in the original hypergraph. This is not a coincidence but a general property of the weighted projection defined by $W_V = BD_{\mathcal{C}}^{-1}B^{\top}$. Moreover, note that G_V and $G_{\mathcal{C}}$ may contains weighted loops.*

Proposition 2.6 *Let $H = (V, \mathcal{C})$ be a hypergraph and let $W_V = BD_{\mathcal{C}}^{-1}B^{\top}$ be its weighted projection onto V. Then, for every vertex $v \in V$, the weighted degree of v in G_V is equal to its degree in the hypergraph: $d_V(v) = d_H(v)$. Equivalently, $W_V\mathbf{1} = \mathbf{d}_V$, where $\mathbf{d}_V$ is the degree vector of the hypergraph. And similarly, for every hyperedge $c \in \mathcal{C}$, the weighted degree of c in $G_{\mathcal{C}}$ is equal to its degree in the hypergraph: $d_{\mathcal{C}}(c) = d_H(c)$. Equivalently, $W_{\mathcal{C}}\mathbf{1} = \mathbf{d}_{\mathcal{C}}$, where $\mathbf{d}_{\mathcal{C}}$ is the degree vector of hyperedge.*

Proof Let $v_i \in V$. By definition, $d_V(v_i) = \sum_j (W_V)_{ij}$. Using $(W_V)_{ij} = \sum_{c\in\mathcal{C}} \frac{B_{ic}B_{jc}}{d_H(c)}$, we obtain $d_V(v_i) = \sum_j \sum_{c\in\mathcal{C}} \frac{B_{ic}B_{jc}}{d_H(c)}$. Interchanging the order of summation gives $d_V(v_i) = \sum_{c\in\mathcal{C}} \frac{B_{ic}}{d_H(c)} \sum_j B_{jc}$. Since $\sum_j B_{jc} = d_H(c)$, it follows that $d_V(v_i) = \sum_{c\in\mathcal{C}} B_{ic}$. The latter quantity is precisely the number of hyperedges containing v_i, namely $d_H(v_i)$. Therefore $d_V(v_i) = d_H(v_i)$, for every vertex $v_i \in V$. □

Notation: We can now write $d(v)$ for v in V and $d(c)$ for c in $\mathcal{C}$ without any ambiguities. And let m be the number of edges of the bipartite graph B, then the sum of degrees of v in V is equal to m and the sum of degrees of c in $\mathcal{C}$ is also equal to m.

3 Modularity and its efficacy in weighted graphs

3.1 Classic modularity formulation

Modularity, introduced by Newman and Girvan [18], is a fundamental metric used to evaluate the quality of a network partition into communities. Given an undirected graph $G = (V, E)$ with $n = |V|$ vertices and $m = |E|$ edges, let $P = \{U_1, U_2, \ldots, U_k\}$ denote a partition of the vertex set V. The Newman's global modularity Q is defined as:

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{d_i d_j}{2m} \right] \delta(U_i, U_j) \tag{3.1}$$

where A is the adjacency matrix of G ($A_{ij} = 1$ if $(i, j) \in E$ and 0 otherwise), $d_i = \sum_j A_{ij}$ is the degree of vertex i, U_i represents the cluster index assigned to vertex i and $\delta(U_i, U_j)$ is the Kronecker delta function.

Alternatively, global modularity can be expressed as the sum of local community contributions $Q(U_r)$:

$$Q = \sum_{r=1}^{k} Q(U_r), \text{ where } Q(U_r) = \frac{e_r}{m} - \left(\frac{d_r}{2m} \right)^2, \tag{3.2}$$

here, $e_r = |\{(u, v) \in E : u, v \in U_r\}|$ denotes the number of intra-community edges in U_r, and $d_r = \sum_{i \in U_r} d_i$ is the total sum of the degree of the vertices in U_r. Higher values of Q indicate stronger partition quality, where the density of internal links significantly exceeds the expected edge densities in a random graph model. In this paper, we will denote by Q_U for the above unweighted modularity.

3.2 Limitation of unweighted modularity: sensitivity to size imbalance

Although widely used, unweighted modularity Q fails to evaluate communities of very different sizes. It underestimates small clusters while over evaluating large ones.

To illustrate this issue, consider two benchmark graph configurations:

Example 3.1 *1. Balanced Graph (G_1): Two symmetric cliques $K_{50} \cup K_{50}$ linked by a single inter-community edge ($m_{inter} = 1$).*

2. Unbalanced Graph (G_2): Two asymmetric cliques $K_{50} \cup K_{10}$ linked by a single inter-community edge ($m_{inter} = 1$).

Recall the standard unweighted modularity formula for a partition $\mathcal{C} = \{U_1, U_2, \ldots, U_k\}$:

$$Q_U = \sum_{r=1}^{k} Q(U_r) = \sum_{r=1}^{k} \left[\frac{e_r}{m} - \left(\frac{d_r}{2m} \right)^2 \right]. \tag{3.3}$$

For the balanced graph G_1, the internal edge counts are $e_1 = e_2 = \binom{50}{2} = 1225$, and the total edge count is $m = 1225 + 1225 + 1 = 2451$. The degree sums are $d_1 = d_2 = 2e_1 + 1 = 2451$. Each

community contributes $Q(U_r) = \frac{1225}{2451} - \left(\frac{2451}{4902}\right)^2 \approx 0.4998 - 0.2500 = 0.2498$*, yielding a high total modularity score* $Q(G_1) \approx 0.4996$.

In contrast, for the unbalanced graph G_2*, the internal edge counts are* $e_1 = \binom{50}{2} = 1225$ *and* $e_2 = \binom{10}{2} = 45$*. With* $m = 1225 + 45 + 1 = 1271$*, the degree sums are* $d_1 = 2e_1 + 1 = 2451$ *and* $d_2 = 2e_2 + 1 = 91$*. Computing the individual contributions to modularity yields:*

$$Q(U_1) = \frac{1225}{1271} - \left(\frac{2451}{2542}\right)^2 \approx 0.9638 - 0.9297 = 0.0341$$
$$Q(U_2) = \frac{45}{1271} - \left(\frac{91}{2542}\right)^2 \approx 0.0354 - 0.0013 = 0.0341$$

resulting in a drastically reduced overall modularity $Q(G_2) \approx 0.0682$.

Notice that both $Q(U_1)$ *and* $Q(U_2)$ *drop close to zero, but for completely different theoretical reasons:*

- *For the large community* U_1*, despite capturing* 96.38% *of all edges in the graph (*$\frac{e_1}{m} \approx 0.9638$*), its contribution* $Q(U_1)$ *is almost entirely wiped out by the quadratic degree penalty* $\left(\frac{d_1}{2m}\right)^2 \approx 0.9297$.
- *For the small, tightly knit clique* U_2*, its contribution* $Q(U_2)$ *decays rapidly because its internal edge proportion* $\frac{e_2}{m} \approx 0.0354$ *is severely diluted by the large volume of the adjacent cluster* U_1.

Lemma 3.2 *Let* G *be an unweighted graph consisting of two clique communities* $U_1 \cong K_{n_1}$ *and* $U_2 \cong K_{n_2}$ *connected by* 1 *inter-community edge, where* $n_1 \geq n_2 \geq 2$*. The total modularity* Q_U *for the natural two-community partition* $\{U_1, U_2\}$ *is given by:*

$$Q_U = 2 \times \frac{n_1(n_1-1)n_2(n_2-1) - 1}{[n_1(n_1-1) + n_2(n_2-1) + 2]^2}. \tag{3.4}$$

Moreover, if $n_1 \gg n_2 \geq 2$*,* Q_U *can be approximated as:*

$$Q_U \approx \frac{n_2^2 - 2n_2}{n_1^2} \sim \mathcal{O}\left(\left(\frac{n_2}{n_1}\right)^2\right). \tag{3.5}$$

Proof By definition, the internal edge counts are $e_1 = \frac{n_1(n_1-1)}{2}$ and $e_2 = \frac{n_2(n_2-1)}{2}$. Since there is 1 inter-community edge, the total number of edges is $m = e_1 + e_2 + 1$.

The degree sums are $d_1 = 2e_1 + 1 = n_1(n_1-1) + 1$ and $d_2 = n_2(n_2-1) + 1$.

$$Q_1 = \frac{n_1(n_1-1)}{n_1(n_1-1) + n_2(n_2-1) + 2} - \left(\frac{n_1(n_1-1) + 1}{n_1(n_1-1) + n_2(n_2-1) + 2}\right)^2$$
$$= \frac{n_1(n_1-1)n_2(n_2-1) - 1}{[n_1(n_1-1) + n_2(n_2-1) + 2]^2}.$$

Idem,

$$Q_2 = \frac{n_1(n_1-1)n_2(n_2-1) - 1}{[n_1(n_1-1) + n_2(n_2-1) + 2]^2}.$$

Then

$$Q_U = Q_1 + Q_2 = 2 \times \frac{n_1(n_1-1)n_2(n_2-1)-1}{[n_1(n_1-1)+n_2(n_2-1)+2]^2} \approx 2 \times \frac{n_2^2 - n_2}{n_1^2}.$$

This completes the proof. □

3.3 Resolution of size imbalance via weighted hypergraph projection

To overcome the size-imbalance resolution bias of unweighted modularity, we construct a weighted graph projection G_V from the hypergraph structure, which naturally normalizes the edge density according to hyperedge sizes. Here, note that although our weighted graph G_V is directed, its symmetry allows us to apply the modularity formula designed for undirected graphs. The only special consideration is how self loops are handled.

Recall that, the following formula for modularity in directed graph was introduced by Leicht and Newman in [16].

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{d_i^{out} d_j^{in}}{2m} \right] \delta(U_i, U_j). \tag{3.6}$$

For our symmetric directed graph, we have weight $w(u,v) = w(v,u)$, the out degree $d_i^{out} = d_i^{in}$, so we can define the following modularity.

Definition 3.3 (Weighted modularity Q_W) *Let $H = (V, \mathcal{C})$ be a hypergraph. Let G_V be its weighted projection graph. The weighted modularity Q_W for a partition $P = \{U_1, U_2, \ldots, U_k\}$ is defined as:*

$$Q_W = \frac{1}{W} \sum_{i,j} \left[w_V(i,j) - \frac{d_W(i) d_W(j)}{W} \right] \delta(U_i, U_j), \tag{3.7}$$

which can be computed by

$$Q_W = \sum_{r=1}^{k} \left[\frac{W_r}{W} - \left(\frac{D_r}{W} \right)^2 \right], \tag{3.8}$$

where $W_r = \sum_{i,j \in U_r} w_V(i, vj$ is the sum of internal edge weights in U_r, $D_r = \sum_{i \in U_r} d_W(i)$ is the sum of the degree of vertices of U_r, and $W = \sum_{\{i,j\} \in E_V} w_V(i,j)$ is the total edge weight in G_V.

3.3.1 Numerical illustration with exact hypergraph projection

Example 3.4 *Let $H = (V, \mathcal{C})$ be a hypergraph with $n = 60$ vertices and hyperedges $\mathcal{C} = \{U_1, U_2, U_3\}$, where $U_1 = \{v_1, \ldots, v_{50}\}$, $U_2 = \{v_{51}, \ldots, v_{60}\}$, and $U_3 = \{v_{50}, v_{51}\}$.*

First, note that the unweigthed projection of H is the graph G_2 in Example 3.1.

Now, lest us consider the weighted projection graph G_V:

- *Internal edges of U_1: each vertex in U_1 has 50 outgoing edges (containing one loop), each edge of weight $\frac{1}{50} = 0.02$, giving $W_1 = 50 \times 50 \times 0.02 = 50$.*
- *Internal edges of U_2: each vertex in U_2 has 10 outgoing edges (containing one loop), each edge of weight $\frac{1}{10} = 0.1$, giving $W_2 = 10 \times 10 \times 0.1 = 10$.*
- *The inter-community edge (v_{50}, v_{51}) and (v_{51}, v_{50}) formed by U_3 has weight $w_{inter} = \frac{1}{2} = 0.5$.*

The total graph weight is $W = W_1 + W_2 + w_{inter} = 61$. *The weighted degree sums are* $D_1 = W_1 + w(v_{50}, v_{51}) = 50.5$ *and* $D_2 = 2W_2 + w(v_{51}, v_{50}) = 10.5$. *Evaluating the weighted modularity terms yields:*

$$Q_W(U_1) = \frac{50}{61} - \left(\frac{50.5}{61}\right)^2 \approx 0.1343$$
$$Q_W(U_2) = \frac{10}{61} - \left(\frac{10.5}{61}\right)^2 \approx 0.1343$$

resulting in a total weighted modularity $Q_W(G_2) \approx 0.2686$.

Remark: *Under the unweighted modularity* Q_U, *both clusters contributed an almost negligible* $Q_U(U_1) \approx Q_U(U_2) \approx 0.0341$ *(total* $Q_U \approx 0.0682$*). By incorporating hypergraph weighted projection,* $Q_W(G_2)$ *increases by more than* 3.9 *times to* 0.2532. *More importantly, the individual contribution of the small clique* U_2 *rises drastically from* 0.0341 *to* 0.1343.

Lemma 3.5 *Let* $H = (V, \mathcal{C})$ *be a hypergraph with two community hyperedges* U_1, U_2 *of sizes* $|U_1| = n_1$ *and* $|U_2| = n_2$ *(*$n_1 \geq n_2 \geq 2$*), and a single inter community hyperedge* $U_3 = \{v_{n_1}, v_{n_1+1}\}$ *of size* $|U_3| = 2$.

The total weighted modularity for the natural two community partition U_1, U_2 *is given by:*

$$Q_W = \frac{2n_1n_2 - 1}{(n_1 + n_2 - 1)^2}. \tag{3.9}$$

Moreover, if $n_1 \gg n_2 \geq 2$, Q_W *scales asymptotically as:*

$$Q_W \sim \mathcal{O}\left(\frac{n_2}{n_1}\right). \tag{3.10}$$

Proof In the weighted projection G_V: The total internal edge weight of clique U_1 is $W_1 = n_1$. The total internal edge weight of clique U_2 is $W_2 = n_2$. The 2 inter-community edge weight from U_3 is $w_{\text{inter}} = \frac{1}{|U_3|} = \frac{1}{2}$.

The total weight volume of G_V is: $W = W_1 + W_2 + w_{\text{inter}} = n_1 + n_2 + 1$.

The degree sum (strength) for each community is: $D_1 = n_1 + 0.5, D_2 = n_2 + 0.5$.

By simple computations, we have:

$$Q_W(U_1) = \frac{W_1}{W} - \left(\frac{D_1}{W}\right)^2 = \frac{n_1}{n_1 + n_2 + 1} - \left(\frac{n_1 + 0.5}{n_1 + n_2 + 1}\right)^2 = \frac{n_1n_2 - 0.5}{(n_1 + n_2 + 1)^2}.$$

And, similarly, $Q_W(U_2) = Q_W(U_1)$, then $Q_W = 2Q_W(U_1)$. This completes the proof. □

3.4 Evaluation under extreme size imbalance ($K_{100} \cup K_{100} \cup K_{10}$)

To investigate the asymptotic behavior with large scale imbalance, we evaluate Q_U and Q_W on G_3, which consists of two massive cliques $C_1 \cong C_2 \cong K_{100}$ and one small clique $C_3 \cong K_{10}$, connected in three pairwise hyperedges of size 2.

3.4.1 Numerical Evaluation of Unweighted Modularity Q_U

In the standard unweighted graph representation of G_3:

- Internal edge counts: $e_1 = \binom{100}{2} = 4950$, $e_2 = \binom{100}{2} = 4950$, and $e_3 = \binom{10}{2} = 45$.
- Inter-community edges: $m_{\text{inter}} = 3$.
- Total edge count: $m = e_1 + e_2 + e_3 + 3 = 4950 + 4950 + 45 + 3 = 9948$.
- Total degree sum: $d_1 = d_2 = 2 \times 4950 + 2 = 9902$, and $d_3 = 2 \times 45 + 2 = 92$.

Evaluating the individual community contributions to unweighted modularity $Q_U(C_r) = \frac{e_r}{m} - \left(\frac{d_r}{2m}\right)^2$ yields:

$$Q_U(C_1) = Q_U(C_2) = \frac{4950}{9948} - \left(\frac{9902}{19896}\right)^2 \approx 0.2499,$$
$$Q_U(C_3) = \frac{45}{9948} - \left(\frac{92}{19896}\right)^2 \approx 0.0045, \text{ which is very small compare to } Q_U(C_1).$$

The global unweighted modularity of $Q_U(G_3) \approx 0.2499 + 0.2499 + 0.0045 = 0.5043$.

3.4.2 Numerical Evaluation of Weighted Modularity Q_W

Under the weighted hypergraph projection G_V with edge weights $w(u, v) = \sum_{c:u,v \in c} \frac{1}{d(c)}$:

- Internal community weights: $W_1 = W_2 = 100 \times 100 \times 0.01 = 100$, and $W_3 = 10$.
- Inter-community weights: Each of the 3 inter-community edges receives weight $w_{\text{inter}} = 0.5$ (in two directions).
- Total graph weight: $W = W_1 + W_2 + W_3 + 3 \times 2 \times 0.5 = 100 + 100 + 10 + 3 = 213$.
- Community strengths: $D_1 = D_2 = 100 + 0.5 + 0.5 = 101$, and $D_3 = 10 + 0.5 + 0.5 = 11$.

Evaluating the weighted modularity contributions $Q_W(C_r) = \frac{W_r}{W} - \left(\frac{D_r}{2W}\right)^2$ yields:

$$Q_W(C_1) = Q_W(C_2) = \frac{100}{213} - \left(\frac{101}{213}\right)^2 \approx 0.2446,$$
$$Q_W(C_3) = \frac{10}{213} - \left(\frac{11}{213}\right)^2 \approx 0.0443,$$

resulting in a total weighted modularity score of $Q_W(G_3) \approx 0.2446 + 0.2446 + 0.0443 \approx 0.5335$.

3.4.3 Comparative Analysis and Key Insights

Table 1 presents the performance metrics comparing Q_U and Q_W on G_3.

We observe that when there are several large clusters, the modularity contribution of the large clusters is considerable, whereas that of the small cluster is lower. Nevertheless, while the modularity of the small cluster in the unweighted version is negligible, its value in the weighted version is substantial enough to enable the recognition of small clusters.

Table 1: Performance evaluation of Q_U and Q_W on an extremely imbalanced network G_{3c} ($K_{100} \cup K_{100} \cup K_{10}$, ratio $k = 10$).

Metric	**Unweighted (Q_U)**	**Weighted (Q_W)**	**Ratio / Advantage**
Total Weight	$m = 9948$	$W = 213$	Weight Rescaling
$Q(C_1), Q(C_2)$ (Massive K_{100})	0.2499	0.2446	Stable Contribution
$Q(C_3)$ (Small K_{10})	0.0045	0.0443	9.84× Gain in Detectability
Global Modularity (Q)	0.5043	0.5336	$Q_W > Q_U$

3.5 Analysis of modularity for (p, q) imbalanced networks

Theorem 3.6 *Let G composed of p large cliques $C_1, \ldots, C_p \cong K_{n_1}$ and q small cliques $C_{p+1}, \ldots, C_{p+q} \cong K_{n_2}$, with $n_1 = k \cdot n_2$ ($k \gg 1$). Each pairwise clique is conneted by an edge. Let consider the partition of G into communities corresponding to its cliques.*

Under proportional scaling where the ratio k is large. The modularity in each community have the following properties:

1. *Large community behavior ($C_{\textbf{large}} \cong K_{n_1}$): Both Q_U and Q_W assign asymptotically equivalent modularity contributions to large communities:*

$$Q_U(C_{large}) \approx \frac{(p-1)k^4 + qk^2}{(pk^2+q)^2} \xrightarrow{k\gg 1} \frac{p-1}{p^2} + \frac{q}{p^2k^2}, \tag{3.11}$$

$$Q_W(C_{large}) \approx \frac{(p-1)k^2 + qk}{(pk+q)^2} \xrightarrow{k\gg 1} \frac{p-1}{p^2} + \frac{q}{p^2k}. \tag{3.12}$$

2. *Small community behavior ($C_{\textbf{small}} \cong K_{n_2}$): Q_U suffers quadratic decay, whereas Q_W degrades only linearly:*

$$Q_U(C_{small}) \approx \frac{pk^2+q-1}{(pk^2+q)^2} \approx \frac{1}{pk^2+q} \approx \frac{1}{pk^2}, \tag{3.13}$$

$$Q_W(C_{small}) \approx \frac{pk+q-1}{(pk+q)^2} \approx \frac{1}{pk+q} \approx \frac{1}{pk}. \tag{3.14}$$

3. *Relative detectability gain: The relative advantage of weighted hypergraph projection modularity Q_W over unweighted modularity Q_U for detecting small communities scales linearly with k:*

$$\lim_{k\to\infty} \left[\frac{\lim_{n_2\to\infty} Q_W(C_{small})}{\lim_{n_2\to\infty} Q_U(C_{small})} \right] = k \tag{3.15}$$

Proof Let m denote the total unweighted edge count and W denote the total sum of weights of edges. As $n_2 \to \infty$:

$$m = p\binom{kn_2}{2} + q\binom{n_2}{2} + \binom{p+q}{2} = \frac{(pk^2+q)n_2^2 - (pk+q)n_2 + (p+q)(p+q-1)}{2} \approx \frac{(pk^2+q)n_2^2}{2}.$$

$$W = pkn_2 + qn_2 + \binom{p+q}{2} \approx (pk+q)n_2.$$

1. Evaluation for C_{large}: For an unweighted large clique $C_i \cong K_{n_1}$ $(1 \le i \le p)$, its internal edge count is $e_i = \binom{kn_2}{2} \approx \frac{k^2 n_2^2}{2}$ and its total degree is $d_i = 2e_i + (p+q-1) \approx k^2 n_2^2$. Therefore:

$$Q_U(C_{\text{large}}) \approx \frac{\frac{k^2 n_2^2}{2}}{\frac{(pk^2+q)n_2^2}{2}} - \left(\frac{k^2 n_2^2}{(pk^2+q)n_2^2}\right)^2 = \frac{k^2}{pk^2+q} - \frac{k^4}{(pk^2+q)^2} = \frac{(p-1)k^4 + qk^2}{(pk^2+q)^2}.$$

Taking $k \gg 1$, dividing the numerator and denominator by k^4 yields $Q_U(C_{\text{large}}) \approx \frac{p-1}{p^2}$.

For the weighted hypergraph projection G_V, $W_{\text{large}} = kn_2$ and $D_{\text{large}} = kn_2 + \frac{p+q-1}{2}$. Thus:

$$Q_W(C_{\text{large}}) \approx \frac{kn_2}{(pk+q)n_2} - \left(\frac{kn_2 + \frac{p+q-1}{2}}{(pk+q)n_2}\right)^2 = \frac{k}{pk+q} - \frac{k^2}{(pk+q)^2} = \frac{(p-1)k^2 + qk}{(pk+q)^2}.$$

Taking $k \gg 1$, dividing numerator and denominator by k^2 also yields $Q_W(C_{\text{large}}) \approx \frac{p-1}{p^2}$. This establishes that Q_W preserves the representation score of large communities without distortion.

2. Evaluation for C_{small}: Under undirected projection, for a small clique $C_j \cong K_{n_2}$ $(p+1 \le j \le p+q)$, $e_j = \binom{n_2}{2} \approx \frac{n_2^2}{2}$ and its degree $d_j \approx n_2^2$:

$$Q_U(C_{\text{small}}) \approx \frac{\frac{n_2^2}{2}}{\frac{(pk^2+q)n_2^2}{2}} - \left(\frac{n_2^2}{(pk^2+q)n_2^2}\right)^2 = \frac{1}{pk^2+q} - \frac{1}{(pk^2+q)^2} \approx \frac{1}{pk^2+q}.$$

Under weighted projection, $W_{\text{small}} = n_2$ and $D_{\text{small}} = n_2 + \frac{p+q-1}{2}$:

$$Q_W(C_{\text{small}}) = \frac{n_2}{(pk+q)n_2} - \left(\frac{n_2 + \frac{p+q-1}{2}}{(pk+q)n_2}\right)^2 \approx \frac{1}{pk+q} - \frac{1}{(pk+q)^2} \approx \frac{1}{pk+q}.$$

Taking $k \gg 1$, we have $Q_U(C_{\text{small}}) \approx \frac{1}{pk^2}$ and $Q_w(C_{\text{small}}) \approx \frac{1}{pk}$ - which is much greater than $Q_U(C_{\text{small}})$.

3. Ratio evaluation: Comparing the asymptotic expressions for C_{small} as $k \gg 1$:

$$\frac{\lim_{n_2\to\infty} Q_W(C_{\text{small}})}{\lim_{n_2\to\infty} Q_U(C_{\text{small}})} = \frac{\frac{pk+q-1}{(pk+q)^2}}{\frac{pk^2+q-1}{(pk^2+q)^2}} \approx \frac{\frac{1}{pk+q}}{\frac{1}{pk^2+q}} = \frac{pk^2+q}{pk+q} \xrightarrow{k\gg 1} k.$$

This completes the proof, confirming that Q_W selectively boosts the signal of small communities by a factor of k while keeping large community scores intact at $\frac{p-1}{p^2}$. □

3.6 Efficiency of weighted modularity

From the examples and analysis in the previous subsection os this section, we can deduce the following properties.

Corollary 3.7 *Let G be a graph containing small and large cliques with a size ratio of k.*

1. *If G contains a single large clique and small cliques, then Q_U is negligible while Q_W remains significant, yielding $\frac{Q_W}{Q_U} \approx k$.*

2. *If G contains multiple large cliques, the modularity values Q_U and Q_W for these large cliques are nearly identical.*

3. *In all cases, for small cliques, Q_U is negligible while Q_W is significant, with $\frac{Q_W}{Q_U} \approx k$.*

We can therefore conclude that applying weighted graphs overcomes key limitations of modularity in unbalanced networks. By assigning meaningful modularity scores to small components, this method improves the discovery of fine grained communities and prevents modularity from vanishing in the presence of a single dominant component. Overall, this yields a more effective and realistic representation.

4 Random walk on projected graphs

Many methods for network analysis are based on random walks. In our setting, random walks naturally arise on the incidence bipartite graph $G_B = (V \cup \mathcal{C}, E_B)$ associated with the hypergraph $H = (V, \mathcal{C})$. At each step, the walk alternates between the two parts of G_B.

Definition 4.1 *Let $G_B = (V \cup \mathcal{C}, E_B)$ be the incidence bipartite graph of the hypergraph $H = (V, \mathcal{C})$. We define a random walk on G_B, also called random walk on H as follows:*

- *From a vertex $v \in V$, the walk moves to a group $c \in \mathcal{C}$ uniformly among all c such that $v \in c$.*
- *From a group $c \in \mathcal{C}$, the walk moves to a vertex $v \in V$ uniformly among all $v \in c$.*

This defines two transition mechanisms:

$$P_{V\mathcal{C}}(v,c) = \begin{cases} \frac{1}{d(v)}, & v \in c \\ 0, & \text{otherwise,} \end{cases} \qquad P_{\mathcal{C}V}(c,v) = \begin{cases} \frac{1}{d(c)}, & v \in c \\ 0, & \text{otherwise.} \end{cases}$$

The full random walk on the bipartite graph can then be written in block matrix form as

$$P_B = \begin{pmatrix} 0 & P_{V\mathcal{C}} \\ P_{\mathcal{C}V} & 0 \end{pmatrix}.$$

In many applications, the primary objects of interest are the vertices in V, while the groups in $\mathcal{C}$ serve only as an intermediate structure describing higher-order interactions. In this case, it is natural to consider a random walk whose state space is restricted to V. Starting from a vertex $u \in V$, the walker first chooses a group $c \in \mathcal{C}$ containing u according to the transition matrix $P_{V\mathcal{C}}$, and then moves from c to a vertex $v \in V$ according to the transition matrix $P_{\mathcal{C}V}$. Thus, one step of the induced walk on V corresponds to two consecutive steps of the bipartite random walk:

Definition 4.2 *The random walk on V induced by the hypergraph $H = (V, \mathcal{C})$ is the Markov chain with transition matrix $P_V = P_{V\mathcal{C}} P_{\mathcal{C}V}$. Equivalently, for any $u, v \in V$, $P_V(u,v) = \sum_{c:\, u,v \in c} \frac{1}{d(u)d(c)}$.*

Since $P_{V\mathcal{C}} = D_V^{-1}B, P_{\mathcal{C}V} = D_{\mathcal{C}}^{-1}B^\top$, the transition matrix can be written in matrix form as $P_V = D_V^{-1}BD_{\mathcal{C}}^{-1}B^\top$.

By duality, one may also consider a random walk whose state space is the set of groups $\mathcal{C}$. In this case, the vertices of V play the role of an intermediate structure.

Definition 4.3 *The random walk on $\mathcal{C}$ induced by the hypergraph $H = (V, \mathcal{C})$ is the Markov chain with transition matrix $P_{\mathcal{C}} = P_{\mathcal{C}V}P_{V\mathcal{C}}$. Equivalently, for any $c, d \in \mathcal{C}$, $P_{\mathcal{C}}(c, d) = \sum_{u\in c\cap d} \frac{1}{d(c)d(u)}$.*

Since $P_{\mathcal{C}V} = D_{\mathcal{C}}^{-1}B^\top, P_{V\mathcal{C}} = D_V^{-1}B$, the transition matrix can be written in matrix form as $P_{\mathcal{C}} = D_{\mathcal{C}}^{-1}B^\top D_V^{-1}B$.

Example 4.4 *We now apply the above construction to the Example hypergraph 2.4. Consider the hypergraph $H = (V, \mathcal{C})$ with $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$, and $\mathcal{C} = \{C_1 = \{1, 2, 3, 4\}, C_2 = \{3, 4, 5\}, C_3 = \{1, 3, 6\}\}$. This yields the transition matrices:*

$$P_{V\mathcal{C}} = \begin{pmatrix} \frac{1}{2} & 0 & \frac{1}{2} \\ 1 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad P_{\mathcal{C}V} = \begin{pmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & 0 \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \\ \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} \end{pmatrix}, \qquad P_B = \begin{pmatrix} 0 & P_{V\mathcal{C}} \\ P_{\mathcal{C}V} & 0 \end{pmatrix}.$$

The induced random walk on V and on $\mathcal{C}$ have transition matrices:

$$P_V = P_{V\mathcal{C}}P_{\mathcal{C}V} = \begin{pmatrix} \frac{7}{24} & \frac{1}{8} & \frac{7}{24} & \frac{1}{8} & 0 & \frac{1}{6} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & 0 \\ \frac{7}{36} & \frac{1}{12} & \frac{1}{3} & \frac{7}{36} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{8} & \frac{1}{8} & \frac{7}{24} & \frac{7}{24} & \frac{1}{6} & 0 \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \\ \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} \end{pmatrix}, \qquad P_{\mathcal{C}} = P_{\mathcal{C}V}P_{V\mathcal{C}} = \begin{pmatrix} \frac{7}{12} & \frac{5}{24} & \frac{5}{24} \\ \frac{5}{18} & \frac{11}{18} & \frac{1}{9} \\ \frac{5}{18} & \frac{1}{9} & \frac{11}{18} \end{pmatrix}.$$

One readily verifies that both P_V and $P_{\mathcal{C}}$ are row-stochastic matrices. They therefore define Markov chains on the vertex set V and on the group set $\mathcal{C}$, respectively. Moreover, they satisfy the following property.

Proposition 4.5 *The random walk induced on V by the bipartite random walk on H coincides with the standard random walk on the weighted projection G_V. Dually, the random walk induced on $\mathcal{C}$ coincides with the standard random walk on the weighted projection $G_{\mathcal{C}}$.*

Proof The transition matrix of the induced random walk on V is $P_V = P_{V\mathcal{C}}P_{\mathcal{C}V} = D_V^{-1}BD_{\mathcal{C}}^{-1}B^\top = D_V^{-1}W_V$. Since the weighted degree matrix of G_V is D_V, the matrix $D_V^{-1}W_V$ is precisely the transition matrix of the standard random walk on G_V.

Similarly, $P_{\mathcal{C}} = P_{\mathcal{C}V}P_{V\mathcal{C}} = D_{\mathcal{C}}^{-1}B^\top D_V^{-1}B = D_{\mathcal{C}}^{-1}W_{\mathcal{C}}$. Since the weighted degree matrix of $G_{\mathcal{C}}$ is $D_{\mathcal{C}}$, this is exactly the transition matrix of the standard random walk on $G_{\mathcal{C}}$. □

Remark 4.6 *This proposition provides a dynamical justification for the weighted projections. The graphs G_V and $G_{\mathcal{C}}$ are not only similarity graphs: their standard random walks reproduce exactly the two-step dynamics in the bipartite random walk of the hypergraph. Consequently, the study of random walks on G_B (on H) can be equivalently carried out on the weighted projections G_V and $G_{\mathcal{C}}$.*

4.1 Stationary distributions

The stationary distributions of the random walks introduced above admit simple closed forms.

Proposition 4.7 *The stationary distribution of the bipartite random walk on G_B is $\pi_H(x) = \frac{d(x)}{2m}, x \in V \cup \mathcal{C}$.*

The stationary distributions of the induced random walks on G_V and $G_\mathcal{C}$ are respectively $\pi_V(v) = \frac{d(v)}{m}, v \in V$, and $\pi_\mathcal{C}(c) = \frac{d(c)}{m}, c \in \mathcal{C}$.

Proof Since P_H is the simple random walk on the undirected bipartite graph $G_B = (V \cup \mathcal{C}, E_B)$ then its stationary distribution is defined by $\pi_H(x) = \frac{d(x)}{\sum_{z \in V \cup \mathcal{C}} d(z)} = \frac{d(x)}{2m}$.

The induced random walks on G_V and $G_\mathcal{C}$ correspond to observing the bipartite walk only when it visits V and $\mathcal{C}$, respectively. Their stationary distributions are therefore obtained by conditioning π_H on the corresponding side: $\pi_V(v) = \frac{\pi_H(v)}{\sum_{u \in V} \pi_H(u)} = \frac{d(v)}{m}$, and similarly $\pi_\mathcal{C}(c) = \frac{\pi_H(c)}{\sum_{d \in \mathcal{C}} \pi_H(d)} = \frac{d(c)}{m}$. □ For the considered example, the degrees of the vertices are $d(v_1) = 2, d(v_2) = 1, d(v_3) = 3, d(v_4) = 2, d(v_5) = 1, d(v_6) = 1$. The degrees of the groups are $d(C_1) = 4, d(C_2) = 3, d(C_3) = 3$.

Since $m = \sum_{v \in V} d(v) = \sum_{c \in \mathcal{C}} d(c) = 10$, the stationary distribution of the induced random walk on V is $\pi_V = \left(\frac{2}{10}, \frac{1}{10}, \frac{3}{10}, \frac{2}{10}, \frac{1}{10}, \frac{1}{10}\right)$, while the stationary distribution of the induced random walk on $\mathcal{C}$ is $\pi_\mathcal{C} = \left(\frac{4}{10}, \frac{3}{10}, \frac{3}{10}\right)$. Finally, ordering the states of the bipartite graph as $(v_1, v_2, v_3, v_4, v_5, v_6, C_1, C_2, C_3)$, the stationary distribution of the bipartite random walk P_H is $\pi_H = \left(\frac{2}{20}, \frac{1}{20}, \frac{3}{20}, \frac{2}{20}, \frac{1}{20}, \frac{1}{20}, \frac{4}{20}, \frac{3}{20}, \frac{3}{20}\right)$.

We observe that vertices with larger participation in groups receive larger stationary probabilities. Similarly, larger groups are visited more frequently by the random walk. These stationary distributions will play a central role in the definition of distances in the next section.

5 Random walk similarity

5.1 Idea of walktrap

In complex network analysis, a fundamental approach to measuring vertex similarity or structural distance is to embed vertices into an Euclidean space $\mathbb{R}^n$ based on random walk processes. Let $G = (V, E)$ be an undirected graph with adjacency matrix A and degree matrix $D = \text{diag}(d_1, d_2, \ldots, d_n)$. The transition matrix P of a random walk on G is defined as $P = D^{-1}A$, where $P_{ij} = P(v_i, v_j)$ represents the probability of moving from vertex v_i to v_j in one step.

The idea of Pons and Latapy [22] is to embed the vertices v_i to the vector $f(v_i) = D^{-\frac{1}{2}} P^t(v_i, .)$, that means $(\frac{P^t(v_i,v_1)}{\sqrt{d_1}}, \cdots, \frac{P^t(v_i,v_n)}{\sqrt{d_n}})$. We can also write $f(v_i) = e_i^T D^{-\frac{1}{2}} P^t$ where e_i denotes the unit vector with 1 in the i-th entry and 0 elsewhere. Let us rewrite their result because we will apply it for later similar proofs. We first state the following technical lemma:

Lemma 5.1 ([22]) *The eigenvalues of the matrix P are real and satisfy:*

$$1 = \lambda_1 > \lambda_2 \geq \cdots \geq \lambda_n > -1$$

Moreover, there exists an orthonormal family of vectors $(s_\alpha)_{1 \leq \alpha \leq n}$ such that each vector $w_\alpha = D^{-\frac{1}{2}} s_\alpha$ and $u_\alpha = D^{\frac{1}{2}} s_\alpha$ are respectively a right and a left eigenvector associated to the eigenvalue

λ_α:

$$\forall \alpha, \quad P w_\alpha = \lambda_\alpha w_\alpha \quad \textit{and} \quad u_\alpha^T P = \lambda_\alpha u_\alpha^T,$$
$$\forall \alpha, \forall \beta, \quad w_\alpha^T u_\beta = \delta_{\alpha\beta}.$$

Proof The matrix P has the same eigenvalues as its similar matrix $\widetilde{P} = D^{\frac{1}{2}} P D^{-\frac{1}{2}} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$. The matrix $\widetilde{P}$ is real and symmetric, so its eigenvalues λ_α are real. The graph G is connected and primitive (the gcd of the cycle lengths of G is 1, due to the loops on each vertex), therefore we can apply the Perron-Frobenius theorem which implies that P has a unique dominant eigenvalue. Therefore we have: $|\lambda_\alpha| < 1$ for $2 \leq \alpha \leq n$.

The symmetry of $\widetilde{P}$ implies that there also exists an orthonormal family s_α of eigenvectors of S satisfying $\forall \alpha, \forall \beta, s_\alpha^T s_\beta = \delta_{\alpha\beta}$ (where $\delta_{\alpha\beta} = 1$ if $\alpha = \beta$ and 0 otherwise). We then directly obtain that the vectors $w_\alpha = D^{-\frac{1}{2}} s_\alpha$ and $u_\alpha = D^{\frac{1}{2}} s_\alpha$ are respectively a right and a left eigenvector of P satisfying $u_\alpha^T w_\beta = \delta_{\alpha\beta}$. Matrix P is a stochastic matrix ($\sum_{j=1}^n P_{ij} = 1$), so its largest eigenvalue is $\lambda_1 = 1$, and the corresponding right eigenvector w has $w_1(i) = w_1(j)$ for all i, j. □

Theorem 5.2 ([22]) *The distance $r^{(t)}(v_i, v_j)$ of $f(v_i)$ and $f(v_j)$ in $\mathbb{R}^n$ is related to the spectral properties of the matrix P by:*

$$(r^{(t)}(v_i, v_j))^2 = \sum_{\alpha=2}^{n} \lambda_\alpha^{2t} \left(w_\alpha(i) - w_\alpha(j)\right)^2, \tag{5.1}$$

where $(\lambda_\alpha)_{1 \leq \alpha \leq n}$ and $(w_\alpha)_{1 \leq \alpha \leq n}$ are respectively the eigenvalues and right eigenvectors of the matrix P.

Proof The above Lemma implies that

$$P = \sum_{\alpha=1}^{n} \lambda_\alpha w_\alpha u_\alpha^T, \quad \text{and} \quad P^t = \sum_{\alpha=1}^{n} \lambda_\alpha^t w_\alpha u_\alpha^T, \quad \text{and so} \quad P^t(v_i, v_j) = \sum_{\alpha=1}^{n} \lambda_\alpha^t w_\alpha(i) u_\alpha^T(j).$$

Now we obtain the expression of the probability vector $P^t(v_i, \cdot)$:

$$P^t(v_i, \cdot) = \sum_{\alpha=1}^{n} \lambda_\alpha^t w_\alpha(i) u_\alpha^T = D^{\frac{1}{2}} \sum_{\alpha=1}^{n} \lambda_\alpha^t w_\alpha(i) s_\alpha^T.$$

So

$$f(v_i) = D^{-\frac{1}{2}} P^t(v_i, \cdot) = \sum_{\alpha=1}^{n} \lambda_\alpha^t w_\alpha(i) s_\alpha^T \quad \text{and} \quad f_\alpha(v_i) = \lambda_\alpha^t w_\alpha(i) = \lambda_\alpha^t \frac{s_\alpha(i)}{\sqrt{d(v_i)}}.$$

Note that $w_1(i) = w_1(j)$ for all i, j. Because $(s_\alpha)_{1 \leq \alpha \leq n}$ is an orthonormal family of vectors, then

$$(r^{(t)}(v_i, v_j))^2 = \|f(v_i) - f(v_j)\|^2 = \sum_{\alpha=2}^{n} \lambda_\alpha^{2t} \left(w_\alpha(i) - w_\alpha(j)\right)^2 = \sum_{\alpha=2}^{n} \lambda_\alpha^{2t} \left(\frac{s_\alpha(i)}{\sqrt{d(v_i)}} - \frac{s_\alpha(j)}{\sqrt{d(v_j)}} \right)^2.$$

□

5.2 Distance on the weighted projected graph G_V

When extending the idea of Walktrap to hypergraphs $H = (V, \mathcal{C})$, a standard approach is to compute random walk distances directly on the weighted projected graph $G_V = (V, E_V)$. The edge weights in G_V are defined by $W_V = BD_{\mathcal{C}}^{-1}B^T$, where B is the $n \times \rfloor$ incidence matrix, and $D_{\mathcal{C}} = \mathrm{diag}(|c_1|, \ldots, |c_\rfloor|)$ contains hyperedge sizes. The transition matrix on G_V is $P_V = D_V^{-1}W_V = D_V^{-1}BD_{\mathcal{C}}^{-1}B^T$, where $D_V = \mathrm{diag}(d(v_1), \ldots, d(v_n))$ is the degree matrix of vertices in V.

Following the standard Walktrap formulation on V, the embedding of V after t steps of random walk on G_V is:

$$\begin{aligned} f_V : V &\longrightarrow \mathbb{R}^n \\ v &\longmapsto D^{-\frac{1}{2}}P_V^t(v, \cdot). \end{aligned}$$

The distance between two vertices $v_i, v_j \in V$ on G_V if determined by

$$r_V^{(t)}(v_i, v_j) = \sqrt{\sum_{k=1}^{n} \frac{\left(P_V^t(v_i, v_k) - P_V^t(v_j, v_k)\right)^2}{d(v_k)}}. \tag{5.2}$$

Let (λ_α) and $(s_{V,\alpha})$ be the eigenvalues and orthonormal eigenvectors of the $n \times n$ symmetric matrix $\widetilde{P}_V = D_V^{\frac{1}{2}}P_V D_V^{-\frac{1}{2}}$. Furthermore, we define $w_\alpha = D_V^{-\frac{1}{2}}s_{V,\alpha}$ and $u_\alpha = D_V^{\frac{1}{2}}s_{V,\alpha}$, which are respectively the right and left eigenvectors of P_V.

Using the method in Subsection 5.1, we have the following result:

Proposition 5.3 *The distance $r^{(t)}(v_i, v_j)$ of $f_V(v_i)$ and $f_V(v_j)$ in $\mathbb{R}^n$ is related to the spectral properties of the matrix P by:*

$$r^{(t)}(v_i, v_j) = \sqrt{\sum_{\alpha=2}^{n} \lambda_\alpha^{2t} \left(w_\alpha(i) - w_\alpha(j)\right)^2} = \sqrt{\sum_{\alpha=2}^{n} \lambda_\alpha^{2t} \left(\frac{s_{V,\alpha}(i)}{\sqrt{d(v_i)}} - \frac{s_{V,\alpha}(j)}{\sqrt{d(v_j)}}\right)^2}. \tag{5.3}$$

Computational complexity on V: Computing the full spectral decomposition of $\widetilde{P}_V$ requires $\mathcal{O}(n^3)$ time. In large hypergraphs where $n \gg \mathbf{c}$, constructing W_V (which easily becomes dense due to large hyperedges) and solving eigenvalue problems on $n \times n$ matrices becomes computationally prohibitive.

5.3 Cross-set distance mapping into $\mathcal{C}$ via $P_{\mathcal{C}}$

To resolve the computational complexity on V, we consider the bipartite representation $B = (V \cup \mathcal{C}, E_B)$. A single transition from V to $\mathcal{C}$ is defined by $P_{V\mathcal{C}} = D_V^{-1}B$. An odd number of steps $(2t+1)$ on B maps a vertex $v_i \in V$ through one step into the space $\mathcal{C}$ then run $2t$ internal steps in $\mathcal{C}$.

We analyze the spectral properties of the projected hyperedge transition matrix $P_{\mathcal{C}} = D_{\mathcal{C}}^{-1}B^T D_V^{-1}B$, whose normalized symmetric matrix is $\widetilde{P}_{\mathcal{C}} = D_{\mathcal{C}}^{-\frac{1}{2}}B^T D_V^{-1}BD_{\mathcal{C}}^{-\frac{1}{2}}$.

Lemma 5.4 *The matrices $\widetilde{P}_V$ and $\widetilde{P}_{\mathcal{C}}$ share exactly the same positive eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_r$, the other eigenvalues are zero. Furthermore, for every orthonormal eigenvector $s_{\mathcal{C},\alpha} \in \mathbb{R}^r$ of $\widetilde{P}_{\mathcal{C}}$ corresponding to $\lambda_\alpha > 0$, the vector $s_{V,\alpha} \in \mathbb{R}^n$ defined by:*

$$s_{V,\alpha} = \frac{1}{\sqrt{\lambda_\alpha}} D_V^{-\frac{1}{2}} B D_{\mathcal{C}}^{-\frac{1}{2}} s_{\mathcal{C},\alpha} \tag{5.4}$$

is a unit eigenvector of $\widetilde{P}_V$ corresponding to the same eigenvalue λ_α.

Proof Let $\widetilde{B} = D_V^{-\frac{1}{2}} B D_{\mathcal{C}}^{-\frac{1}{2}}$. Then $\widetilde{P}_V = \widetilde{B}\widetilde{B}^T$ and $\widetilde{P}_{\mathcal{C}} = \widetilde{B}^T\widetilde{B}$, so they have only non negative eigenvalues. By definition, $\widetilde{P}_{\mathcal{C}} s_{\mathcal{C},\alpha} = \lambda_\alpha s_{\mathcal{C},\alpha} \implies \widetilde{B}^T\widetilde{B} s_{\mathcal{C},\alpha} = \lambda_\alpha s_{\mathcal{C},\alpha}$. Multiplying both sides from the left by $\widetilde{B}$, we obtain:

$$\widetilde{B}\widetilde{B}^T(\widetilde{B} s_{\mathcal{C},\alpha}) = \lambda_\alpha(\widetilde{B} s_{\mathcal{C},\alpha}) \implies \widetilde{P}_V(\widetilde{B} s_{\mathcal{C},\alpha}) = \lambda_\alpha(\widetilde{B} s_{\mathcal{C},\alpha}).$$

Thus, $s_{V,\alpha} = \frac{1}{\sqrt{\lambda_\alpha}}\widetilde{B} s_{\mathcal{C},\alpha}$ is an eigenvector of $\widetilde{P}_V$ with eigenvalue λ_α. Verifying the norm:

$$\|\widetilde{B} s_{\mathcal{C},\alpha}\|^2 = s_{\mathcal{C},\alpha}^T \widetilde{B}^T \widetilde{B} s_{\mathcal{C},\alpha} = s_{\mathcal{C},\alpha}^T(\lambda_\alpha s_{\mathcal{C},\alpha}) = \lambda_\alpha \|s_{\mathcal{C},\alpha}\|^2 = \lambda_\alpha.$$

Dividing by $\sqrt{\lambda_\alpha}$ yields $\|s_{V,\alpha}\| = 1$, completing the proof. □

Theorem 5.5 *The embedding vector of vertex $v_i \in V$ mapped into the hyperedge space $\mathbb{R}^r$ after $(2t+1)$ steps on G_B is $f_{\mathcal{C}}(v_i) = \mathbf{e}_i^T P_B^{2t+1} D_{\mathcal{C}}^{-\frac{1}{2}}$. Its k-th coordinate relative to the spectral basis $\{s_{\mathcal{C},\alpha}\}$ of $\widetilde{P}_{\mathcal{C}}$ is given by:*

$$f_{\mathcal{C},\alpha}(v_i) = \lambda_\alpha^t \frac{(B D_{\mathcal{C}}^{-\frac{1}{2}} s_{\mathcal{C},\alpha})_i}{d(v_i)}, \tag{5.5}$$

and the resulting Walktrap distance between $f_{\mathcal{C}}(v_i)$ and $f_{\mathcal{C}}(v_j)$ in $\mathbb{R}^r$ is:

$$r_{V\to C}^{(t)}(v_i, v_j) = \sqrt{\sum_{\alpha=2}^{r} \lambda_\alpha^{2t} \left(\frac{(B D_{\mathcal{C}}^{-\frac{1}{2}} s_{\mathcal{C},\alpha})_i}{d(v_i)} - \frac{(B D_{\mathcal{C}}^{-\frac{1}{2}} s_{\mathcal{C},\alpha})_j}{d(v_j)} \right)^2}. \tag{5.6}$$

Proof

The normalized symmetric operator $\widetilde{P}_{\mathcal{C}} = D_{\mathcal{C}}^{\frac{1}{2}} P_{\mathcal{C}} D_{\mathcal{C}}^{-\frac{1}{2}} = D_{\mathcal{C}}^{-\frac{1}{2}} B^T D_V^{-1} B D_{\mathcal{C}}^{-\frac{1}{2}}$ admits the spectral decomposition:

$$\widetilde{P}_{\mathcal{C}}^t = \sum_{\alpha=1}^{r} \lambda_\alpha^t s_{\mathcal{C},\alpha} s_{\mathcal{C},\alpha}^T. \tag{5.7}$$

The embedding vector of vertex $v_i \in V$ into the hyperedge space after $(2t+1)$ steps on G_B is defined as:

$$f_{\mathcal{C}}(v_i) = \mathbf{e}_i^T P_{V\to\mathcal{C}} P_{\mathcal{C}}^t D_{\mathcal{C}}^{-\frac{1}{2}} = \mathbf{e}_i^T D_V^{-1} B \left(D_{\mathcal{C}}^{-\frac{1}{2}} \widetilde{P}_{\mathcal{C}}^t D_{\mathcal{C}}^{\frac{1}{2}} \right) D_{\mathcal{C}}^{-\frac{1}{2}} = \mathbf{e}_i^T D_V^{-1} B D_{\mathcal{C}}^{-\frac{1}{2}} \sum_{l=1}^{r} \lambda_l^t s_{\mathcal{C},l} s_{\mathcal{C},l}^T. \tag{5.8}$$

Projecting $f_{\mathcal{C}}(v_i)$ onto the k-th orthonormal spectral basis vector $s_{\mathcal{C},\alpha}$ yields its k-th coordinate:

$$f_{\mathcal{C},\alpha}(v_i) = f_{\mathcal{C}}(v_i) s_{\mathcal{C},\alpha} = \mathbf{e}_i^T D_V^{-1} B D_{\mathcal{C}}^{-\frac{1}{2}} \left(\sum_{l=1}^{r} \lambda_l^t s_{\mathcal{C},l} s_{\mathcal{C},l}^T \right) s_{\mathcal{C},\alpha}. \tag{5.9}$$

By the orthonormality condition $s_{\mathcal{C},l}^T s_{\mathcal{C},\alpha} = \delta_{l\alpha}$, all terms for $l \neq k$ vanish, simplifying the expression to:

$$f_{\mathcal{C},\alpha}(v_i) = \lambda_\alpha^t \, \mathbf{e}_i^T D_V^{-1} B D_{\mathcal{C}}^{-\frac{1}{2}} s_{\mathcal{C},\alpha} = \lambda_\alpha^t \frac{(B D_{\mathcal{C}}^{-\frac{1}{2}} s_{\mathcal{C},\alpha})_i}{d(v_i)}. \tag{5.10}$$

Taking the L_2 norm $\|f_{\mathcal{C}}(v_i) - f_{\mathcal{C}}(v_j)\|$ over the orthonormal basis $\{s_{\mathcal{C},\alpha}\}$ yields $r_{V\to C}^{(t)}(v_i, v_j)$. □ **Computational complexity:** Since $\widetilde{P}_{\mathcal{C}}$ has dimension $\mathbf{c} \times \mathbf{c}$, its spectral decomposition requires $\mathcal{O}(\mathbf{c}^3)$ time. Transforming eigenvectors to V requires a sparse matrix-vector multiplication $Bs_{\mathcal{C},\alpha}$, taking $\mathcal{O}(|B|) = \mathcal{O}(n\mathbf{c})$ per component. The overall complexity is $\mathcal{O}(\mathbf{c}^3 + r\mathcal{O}(n\mathbf{c})$, which is considerably faster than $\mathcal{O}(n^3|)$ when $\mathbf{c} \ll n$.

5.4 Distance on V via P_V accelerated by $P_{\mathcal{C}}$ spectrum

We now return to the original vertex-to-vertex Walktrap distance $r_V^{(t)}$ on G_V defined in Subsection 5.2, and formulate an accelerated computation leveraging the spectrum of $P_{\mathcal{C}}$.

Theorem 5.6 *The vertex embedding vector $f_V(v_i) \in \mathbb{R}^n$ on G_V can be expressed along the k-th spectral dimension using only the eigensystem $(\lambda_\alpha, s_{\mathcal{C},\alpha})$ of $\widetilde{P}_{\mathcal{C}}$:*

$$f_{V,\alpha}(v_i) = \lambda_\alpha^{t-\frac{1}{2}} \frac{(B D_{\mathcal{C}}^{-\frac{1}{2}} s_{\mathcal{C},\alpha})_i}{d(v_i)}. \tag{5.11}$$

Consequently, the exact vertex-to-vertex Walktrap distance $r_V^{(t)}(v_i, v_j)$ on G_V is evaluated as:

$$r_V^{(t)}(v_i, v_j) = \sqrt{\sum_{\alpha=2}^{r} \lambda_\alpha^{2t-1} \left(\frac{(B D_{\mathcal{C}}^{-\frac{1}{2}} s_{\mathcal{C},\alpha})_i}{d(v_i)} - \frac{(B D_{\mathcal{C}}^{-\frac{1}{2}} s_{\mathcal{C},\alpha})_j}{d(v_j)} \right)^2}. \tag{5.12}$$

Proof Recall from Subsection 1 that $f_{V,\alpha}(v_i) = \lambda_\alpha^t \frac{s_{V,\alpha}(i)}{\sqrt{d(v_i)}}$. Substituting the eigenvector identity $s_{V,\alpha} = \frac{1}{\sqrt{\lambda_\alpha}} D_V^{-\frac{1}{2}} B D_{\mathcal{C}}^{-\frac{1}{2}} s_{\mathcal{C},\alpha}$ from Lemma 5.4, we have:

$$f_{V,\alpha}(v_i) = \lambda_\alpha^t \frac{1}{\sqrt{d(v_i)}} \left(\frac{1}{\sqrt{\lambda_\alpha}} \frac{(B D_{\mathcal{C}}^{-\frac{1}{2}} s_{\mathcal{C},\alpha})_i}{\sqrt{d(v_i)}} \right) = \lambda_\alpha^{t-\frac{1}{2}} \frac{(B D_{\mathcal{C}}^{-\frac{1}{2}} s_{\mathcal{C},\alpha})_i}{d(v_i)}$$

Squaring the difference of coordinates $(f_{V,\alpha}(v_i) - f_{V,\alpha}(v_j))^2$ and summing over $\alpha = 1, \ldots, r$ yields the exponent $(2t-1)$ for λ_α, completing the proof. □

Complexity analysis and comparison: By applying Theorem 5.6, we completely avoid the explicit construction of W_V and the eigen decomposition of the large $n \times n$ matrix $\widetilde{P}_V$.

- Eigensolving: Performed on matrix $\mathbf{c} \times \mathbf{c}$ in $\mathcal{O}(\mathbf{c}^3)$ time.
- Projection to V: Performed via sparse matrix-vector multiplication $B(D_{\mathcal{C}}^{-\frac{1}{2}} s_{\mathcal{C},\alpha})$ in $\mathcal{O}(n\mathbf{c})$ time.

This dual-space formulation reduces the overall execution time from $\mathcal{O}(n^3)$ down to $\mathcal{O}(\mathbf{c}^3 + rn\mathbf{c})$, making the exact computation of Q_W and spectral random walk similarity scalable to massive hypergraphs.

5.5 Hitting time-based random walk distance

An efficient approach to defining the distance between two graph vertices is to embed the graph into the Euclidean space $\mathbb{R}^N$ using hitting times. Specifically, rather than embedding a vertex v_i into the transition probability vector P^t after t steps, we employ the vector $h(v_i, \cdot)$ representing the expected hitting time from v_i to all other vertices. This methodology was introduced by Dang, Do and Phan [7], and we present its adaptation here for our edge-weighted graph framework.

The *Hitting Time* $h(v_i, v_j)$ (or expected first-passage time) from vertex v_i to vertex v_j is the expected number of steps a random walk starting at v_i takes to reach v_j for the first time:

$$h(v_i, v_j) = \mathbb{E}\left[\min\{t \geq 0 \mid X_t = v_j\} \mid X_0 = v_i\right]. \tag{5.13}$$

with $h(v_i, v_i) = 0$ for all $v_i \in V$.

Let $\pi = (\pi_1, \pi_2, \ldots, \pi_n)^T$ be the stationary distribution vector with $\pi_k = \frac{d(v_k)}{m}$.

We define the *Hitting Time Embedding* of a vertex v_i as a vector in $\mathbb{R}^n$ as the following:

$$h(v_i) = \left(\sqrt{\pi_1}h(v_i, v_1), \sqrt{\pi_2}h(v_i, v_2), \ldots, \sqrt{\pi_n}h(v_i, v_n)\right)^T. \tag{5.14}$$

Consequently, the structural distance $r_H(v_i, v_j)$ between two vertices v_i and v_j is evaluated as the Euclidean distance between their corresponding hitting time profile vectors $h(v_i)$ and $h(v_j)$ in $\mathbb{R}^n$:

$$r_H(v_i, v_j) = \|h(v_i) - h(v_j)\|_2 = \sqrt{\sum_{k=1}^{n} \pi_k (h(v_i, v_k) - h(v_j, v_k))^2}. \tag{5.15}$$

5.5.1 Representation via the fundamental matrix Z

To compute $h(v_i, v_j)$ efficiently without step-by-step simulations, we employ the *Fundamental Matrix* Z of the finite Markov chain [26]. Let $\mathbf{\Pi} = \mathbf{1}\pi^T$ be the stationary transition matrix (where $\mathbf{1}$ is the all - ones column vector).

The fundamental matrix $Z = (z_{ij})_{n \times n}$ is defined as:

$$Z = (I - P + \mathbf{\Pi})^{-1} = I + \sum_{m=1}^{\infty} (P^m - \mathbf{\Pi}). \tag{5.16}$$

As proven by Takács [26], the expected hitting time $h(v_i, v_j)$ can be expressed directly in terms of the elements of Z:

$$h(v_i, v_j) = \frac{z_{jj} - z_{ij}}{\pi_j}. \tag{5.17}$$

So we can write the distance as a fonction on the matrix Z:

Lemma 5.7 *The distance based on hitting time can be computed by:*

$$r_H(v_i, v_j) = \sqrt{m \sum_{k=1}^{n} \frac{(z_{jk} - z_{ik})^2}{d(v_k)}}. \tag{5.18}$$

5.5.2 Spectral decomposition of Z and spectral coordinates

We will use an important result on the matrix Z in [26] (using the above notations of matrix P_V, its right and left eigenvectors $w\alpha$ and u_α).

Theorem 5.8 *[26] The fundamental matrix Z can be computed as follows:*

$$Z = \sum_{\alpha=1}^{r} \frac{1}{1-\lambda_\alpha} w_\alpha u_\alpha^T. \tag{5.19}$$

By applying the same argument as in Theorem 5.2, we can now state the result on the distance using hitting time.

Theorem 5.9 *The distance r between $h(v_i)$ and $h(v_j)$ is related to the spectral of P_V by:*

$$(r_H(v_i, v_j))^2 = \sum_{\alpha=2}^{r} m \frac{(w_\alpha(i) - w_\alpha(j))^2}{(1-\lambda_\alpha)^2} = \sum_{\alpha=2}^{r} \frac{m}{(1-\lambda_\alpha)^2} \left(\frac{s_{V,\alpha}(i)}{\sqrt{d(v_i)}} - \frac{s_{V,\alpha}(j)}{\sqrt{d(v_j)}} \right)^2. \tag{5.20}$$

Accelerated hitting time distance via $P_{\mathcal{C}}$ spectrum The distance defined by hitting time can be efficiently computed by using the spectrum of matrix $P_{\mathcal{C}}$. Recall the formula of Theorem 5.9:

$$(r_H(v_i, v_j))^2 == \sum_{\alpha=2}^{r} \frac{m}{(1-\lambda_\alpha)^2} \left(\frac{s_{V,\alpha}(i)}{\sqrt{d(v_i)}} - \frac{s_{V,\alpha}(j)}{\sqrt{d(v_j)}} \right)^2, \tag{5.21}$$

and using the formula 5.4

$$s_{V,\alpha} = \frac{1}{\sqrt{\lambda_\alpha}} D_V^{-\frac{1}{2}} B D_{\mathcal{C}}^{-\frac{1}{2}} s_{\mathcal{C},\alpha}, \tag{5.22}$$

we have:

Theorem 5.10 *The vertex-to-vertex Hitting Time distance $r_{H,V}(v_i, v_j)$ on G_V is given by:*

$$r_H(v_i, v_j) = \sqrt{\sum_{\alpha=2}^{r} \frac{1}{\lambda_\alpha (1-\lambda_\alpha)^2} \left(\frac{(B D_{\mathcal{C}}^{-\frac{1}{2}} s_{\mathcal{C},\alpha})_i}{d(v_i)} - \frac{(B D_{\mathcal{C}}^{-\frac{1}{2}} s_{\mathcal{C},\alpha})_j}{d(v_j)} \right)^2} \tag{5.23}$$

Spectral sensitivity advantage (comparison with Walktrap):

- In the Walktrap metric $r^{(t)}$, the weight assigned to the k-th spectral component is λ_α^{2t}. If two leading eigenvalues are very close to 1, say $\lambda_2 = 0.98$ and $\lambda_3 = 0.95$, their relative weight ratio after $t = 2$ steps is $(0.98/0.95)^4 \approx 1.13$, representing only a modest increase in contrast.
- In contrast, the Hitting Time metric assigns the weight factor $\frac{1}{(1-\lambda_\alpha)^2}$. For the same eigenvalues, the weight factor for λ_2 is $\frac{1}{(1-0.98)^2} = 2500$, whereas for λ_3 it is $\frac{1}{(1-0.95)^2} = 400$. The weight ratio leaps to $\frac{2500}{400} = 6.25$.

This property is particularly important in many applications: when computing vertex distances or evaluating spectral projections, interest often focuses solely on the first few dominant eigenvalues. For instance, although the distance formula formally sums over λ_α from 2 to r, in practice one often considers only the leading terms λ_2. In particular, when λ_2 and λ_3 are very close, relying solely on λ_α^2 fails to distinguish them adequately. In contrast, the weighting factor $\frac{1}{(1-\lambda_\alpha)^2}$ substantially presents the gap between λ_2 and λ_3, providing. So the much sharper resolution. So, the hitting time distance is significantly more sensitive to dominant structural bottlenecks than t-step random walks.

Complexity analysis and efficiency comparison:

- **Computational cost on V:** Exact matrix inversion or solving the fundamental matrix $(I - P_V)^{-1}$ on G_V incurs an overwhelming time complexity of $\mathcal{O}(n^3)$. Furthermore, computing the full pseudo-inverse of the graph Laplacian on dense $n \times n$ projection matrices demands $\mathcal{O}(n^2)$ memory storage, rendering it intractable for massive hypergraph vertex sets.
- **Direct method on G_V:** Requires computing pseudo-inverses or full eigensystems on $n \times n$ matrices with time complexity $\mathcal{O}(n^3)$ and memory footprint $\mathcal{O}(n^2)$.
- **Proposed $P_\mathcal{C}$ accelerated method:**
  1. Solving the eigensystem of $\widetilde{P}_\mathcal{C}$ takes $\mathcal{O}(\mathbf{c}^3)$ time.
  2. Mapping eigenvectors to V via sparse matrix multiplication $B(D_\mathcal{C}^{-\frac{1}{2}} s_{\mathcal{C},\alpha})$ requires $\mathcal{O}(n\mathbf{c})$ time per component.

Overall execution time is reduced to $\mathcal{O}(\mathbf{c}^3 + n\mathbf{c})$. This dual-space spectral transformation makes high-sensitivity Hitting Time distance computation fully scalable for hypergraphs where $\mathbf{c} \ll n$.

5.6 Numerical example: spectral distance and hitting time computations

To demonstrate the efficiency and consistency of our spectral reduction framework via $P_\mathcal{C}$, we consider a hypergraph $H = (V, \mathcal{C})$ with $n = 6$ vertices and $\rfloor = 3$ hyperedges:

$$V = \{v_1, \ldots, v_6\}, \quad \mathcal{C} = \{C_1 = \{1,2,3,4\}, C_2 = \{3,4,5\}, C_3 = \{1,3,6\}\}.$$

The corresponding degree matrices are $D_V = \mathrm{diag}(2,1,3,2,1,1)$ and $D_\mathcal{C} = \mathrm{diag}(4,3,3)$.

5.6.1 Eigensystem of $P_\mathcal{C}$

Instead of diagonalizing the 6×6 matrix P_V, we compute the spectral decomposition of the 3×3 normalized hyperedge matrix $\widetilde{P}_\mathcal{C} = D_\mathcal{C}^{-\frac{1}{2}} B^T D_V^{-1} B D_\mathcal{C}^{-\frac{1}{2}}$:

$$\widetilde{P}_\mathcal{C} = \begin{bmatrix} \frac{7}{12} & \frac{5}{12\sqrt{3}} & \frac{5}{12\sqrt{3}} \\ \frac{5}{12\sqrt{3}} & \frac{11}{18} & \frac{1}{9} \\ \frac{5}{12\sqrt{3}} & \frac{1}{9} & \frac{11}{18} \end{bmatrix}.$$

The non-zero positive eigenvalues and their corresponding orthonormal eigenvectors $s_{\mathcal{C},\alpha}$ are:

$$\lambda_1 = 1, \qquad s_{\mathcal{C},1} = \left[\tfrac{2}{\sqrt{10}}, \quad \tfrac{\sqrt{3}}{\sqrt{10}}, \quad \tfrac{\sqrt{3}}{\sqrt{10}}\right]^T,$$
$$\lambda_2 = \frac{1}{2} = 0.5, \qquad s_{\mathcal{C},2} = \left[0, \quad -\tfrac{1}{\sqrt{2}}, \quad \tfrac{1}{\sqrt{2}}\right]^T,$$
$$\lambda_3 = \frac{11}{36} \approx 0.30556, \qquad s_{\mathcal{C},3} = \left[-\tfrac{\sqrt{3}}{\sqrt{5}}, \quad \tfrac{1}{\sqrt{10}}, \quad \tfrac{1}{\sqrt{10}}\right]^T.$$

We evaluate the projected spectral basis elements $\phi_{\alpha,i} = \frac{(BD_{\mathcal{C}}^{-\frac{1}{2}} s_{\mathcal{C},\alpha})_i}{d(v_i)}$ for each vertex $v_i \in V$ in Table 2. Note that for $\alpha = 1$, $\phi_{1,i} = \frac{1}{\sqrt{10}} \approx 0.31623$ is constant across all vertices.

Vertex v_i	$d(v_i)$	$\phi_{1,i}$ $(\alpha = 1)$	$\phi_{2,i}$ $(\alpha = 2)$	$\phi_{3,i}$ $(\alpha = 3)$
v_1	2	$1/\sqrt{10} \approx 0.31623$	$\sqrt{6}/12 \approx 0.20412$	$-\frac{\sqrt{15}}{20} + \frac{\sqrt{30}}{60} \approx -0.10236$
v_2	1	$1/\sqrt{10} \approx 0.31623$	0.00000	$-\frac{\sqrt{15}}{10} \approx -0.38730$
v_3	3	$1/\sqrt{10} \approx 0.31623$	0.00000	$-\frac{\sqrt{15}}{30} + \frac{\sqrt{30}}{45} \approx -0.00738$
v_4	2	$1/\sqrt{10} \approx 0.31623$	$-\sqrt{6}/12 \approx -0.20412$	$-\frac{\sqrt{15}}{20} + \frac{\sqrt{30}}{60} \approx -0.10236$
v_5	1	$1/\sqrt{10} \approx 0.31623$	$-\sqrt{6}/6 \approx -0.40825$	$\sqrt{30}/30 \approx 0.18257$
v_6	1	$1/\sqrt{10} \approx 0.31623$	$\sqrt{6}/6 \approx 0.40825$	$\sqrt{30}/30 \approx 0.18257$

Table 2: Spectral basis elements $\phi_{\alpha,i} = \frac{(BD_{\mathcal{C}}^{-\frac{1}{2}} s_{\mathcal{C},\alpha})_i}{d(v_i)}$ for $v_i \in V$.

5.6.2 Distance computations and formulations

We compute three distinct distance metrics leveraging the eigensystem of $P_{\mathcal{C}}$ at $t = 2$:

1. **Vertex Walktrap distance $r_V^{(2)}(v_i, v_j)$ at $t = 2$ (Theorem 5.6):**

$$r_V^{(2)}(v_i, v_j) = \sqrt{\sum_{\alpha=2}^{3} \lambda_\alpha^{2t-1} \left(\phi_{\alpha,i} - \phi_{\alpha,j}\right)^2} = \sqrt{(0.5)^3(\phi_{2,i} - \phi_{2,j})^2 + \left(\frac{11}{36}\right)^3 (\phi_{3,i} - \phi_{3,j})^2}.$$

2. **Cross-set Walktrap distance $r_{V\to\mathcal{C}}^{(2)}(v_i, v_j)$ at $t = 2$ (Theorem 5.5):**

$$r_{V\to\mathcal{C}}^{(2)}(v_i, v_j) = \sqrt{\sum_{\alpha=2}^{3} \lambda_\alpha^{2t} \left(\phi_{\alpha,i} - \phi_{\alpha,j}\right)^2} = \sqrt{(0.5)^4(\phi_{2,i} - \phi_{2,j})^2 + \left(\frac{11}{36}\right)^4 (\phi_{3,i} - \phi_{3,j})^2}.$$

3. **Commute hitting time distance $r_H(v_i, v_j)$ via $P_{\mathcal{C}}$ spectrum (Theorem 5.10):** Hitting time distances replace step-based power decay λ_α^t with the Green's function scaling $\frac{1}{\lambda_\alpha(1-\lambda_\alpha)^2}$:

$$r_H(v_i, v_j) = \sqrt{\sum_{\alpha=2}^{3} \frac{1}{\lambda_\alpha(1-\lambda_\alpha)^2} \left(\phi_{\alpha,i} - \phi_{\alpha,j}\right)^2}.$$

For $\alpha = 2$, the factor is $\frac{1}{0.5(1-0.5)^2} = 8$. For $\alpha = 3$, the factor is $\frac{1}{\frac{11}{36}(1-\frac{11}{36})^2} = \frac{46656}{6875} \approx 6.7863$.

Metric	Pair	v_1	v_2	v_3	v_4	v_5	v_6
$r_V^{(2)}$	v_1	**0.0000**	0.0898	0.0717	0.1443	0.2319	0.1020
	v_2	0.0898	**0.0000**	0.0981	0.0898	0.2037	0.2037
	v_3	0.0717	0.0981	**0.0000**	0.0717	0.1557	0.1557
	v_4	0.1443	0.0898	0.0717	**0.0000**	0.1020	0.2319
	v_5	0.2319	0.2037	0.1557	0.1020	**0.0000**	0.2887
	v_6	0.1020	0.2037	0.1557	0.2319	0.2887	**0.0000**
$r_{V\to\mathcal{C}}^{(2)}$	v_1	**0.0000**	0.0636	0.0507	0.1021	0.1639	0.0721
	v_2	0.0636	**0.0000**	0.0694	0.0636	0.1441	0.1441
	v_3	0.0507	0.0694	**0.0000**	0.0507	0.1101	0.1101
	v_4	0.1021	0.0636	0.0507	**0.0000**	0.0721	0.1639
	v_5	0.1639	0.1441	0.1101	0.0721	**0.0000**	0.2041
	v_6	0.0721	0.1441	0.1101	0.1639	0.2041	**0.0000**
r_H	v_1	**0.0000**	0.9404	0.6281	1.1547	1.8844	0.9404
	v_2	0.9404	**0.0000**	0.9897	0.9404	1.8808	1.8808
	v_3	0.6281	0.9897	**0.0000**	0.6281	1.2563	1.2563
	v_4	1.1547	0.9404	0.6281	**0.0000**	0.9404	1.8844
	v_5	1.8844	1.8808	1.2563	0.9404	**0.0000**	2.3094
	v_6	0.9404	1.8808	1.2563	1.8844	2.3094	**0.0000**

Table 3: Comparison of pair-wise distance matrices across metrics at $t = 2$.

5.6.3 Comparative Analysis and Structural Observations

Comparing the distance values across the three methods in Table 3 yields several key insights:

1. **Strict rank preservation:** All three distance metrics preserve the exact relative structural order among vertex pairs. Specifically:

$$r(v_1, v_3) < r(v_1, v_2) < r(v_4, v_5) < r(v_1, v_4) < r(v_1, v_5) < r(v_5, v_6).$$

   The pairs (v_1, v_3) and (v_1, v_2) consistently exhibit smaller distances because they share hyperedges C_1 and C_3, with v_3 acting as the structural hub. Conversely, (v_5, v_6) achieves the maximum distance since $v_5 \in C_2$ and $v_6 \in C_3$ are located at structurally opposite ends of the hypergraph.

2. **Computational advantage:** Calculating these distance matrices via $P_{\mathcal{C}}$ required diagonalizing only a 3×3 matrix instead of a 6×6 matrix. This highlights the practical scalability of computing vertex distances through hyperedge space when $\rfloor \ll n$.

5.6.4 Hypergraph Topology and Eigensystem

We now consider a larger example to examine the effect of using only λ in the distance formula. The vertex set V is distributed across four clusters:

$$|C_1| = 1500, \quad |C_2| = 1500, \quad |C_3| = 1000, \quad |C_4| = 1000.$$

The hyperedge collection $\mathcal{C} = \{e_1, e_2, e_3, e_4, e_{\text{bridge}}\}$ consists of four intra-community hyperedges covering their respective clusters, and one heterogeneous bridge hyperedge e_{bridge} containing:

- 20 vertices from C_1 and 20 vertices from C_2 (forming a narrow macro-bottleneck),
- 300 vertices from C_3 and 300 vertices from C_4 (forming a wider micro-bottleneck).

By applying our dual cell-space reduction, we compute the spectral decomposition of the 5×5 normalized hyperedge transition matrix $\widetilde{P}_{\mathcal{C}} = D_{\mathcal{C}}^{-\frac{1}{2}} B^T D_V^{-1} B D_{\mathcal{C}}^{-\frac{1}{2}}$ rather than diagonalizing a 5000×5000 vertex matrix P_V. The explicit matrix $\widetilde{P}_{\mathcal{C}}$ is evaluated as:

$$\widetilde{P}_{\mathcal{C}} = \begin{bmatrix} 0.9867 & 0.0000 & 0.0000 & 0.0000 & 0.1147 \\ 0.0000 & 0.9867 & 0.0000 & 0.0000 & 0.1147 \\ 0.0000 & 0.0000 & 0.7000 & 0.0000 & 0.5196 \\ 0.0000 & 0.0000 & 0.0000 & 0.7000 & 0.5196 \\ 0.1147 & 0.1147 & 0.5196 & 0.5196 & 0.4133 \end{bmatrix}.$$

Diagonalizing $\widetilde{P}_{\mathcal{C}}$ yields the non-zero positive eigenvalues λ_α and their corresponding orthonormal eigenvectors $s_{\mathcal{C},\alpha}$:

$$\begin{aligned}
&\lambda_1 = 1.0000, && s_{\mathcal{C},1} = \begin{bmatrix} 0.5283, & 0.5283, & 0.4313, & 0.4313, & 0.2743 \end{bmatrix}^T, \\
&\lambda_2 = 0.9867, && s_{\mathcal{C},2} = \begin{bmatrix} \frac{1}{\sqrt{2}}, & -\frac{1}{\sqrt{2}}, & 0, & 0, & 0 \end{bmatrix}^T = \begin{bmatrix} 0.7071, & -0.7071, & 0, & 0, & 0 \end{bmatrix}^T, \\
&\lambda_3 = 0.7000, && s_{\mathcal{C},3} = \begin{bmatrix} 0, & 0, & \frac{1}{\sqrt{2}}, & -\frac{1}{\sqrt{2}}, & 0 \end{bmatrix}^T = \begin{bmatrix} 0, & 0, & 0.7071, & -0.7071, & 0 \end{bmatrix}^T, \\
&\lambda_4 = 0.4133, && s_{\mathcal{C},4} = \begin{bmatrix} -0.2789, & -0.2789, & 0.5485, & 0.5485, & -0.5052 \end{bmatrix}^T.
\end{aligned}$$

The non-zero eigenvalues and their spectral power factors at $t = 2$ are summarized in Table 4.

α	Eigenvalue λ_α	Walktrap λ_α^3 $(r_V^{(2)})$	Walktrap λ_α^4 $(r_{V\to\mathcal{C}}^{(2)})$	Hitting Time $\frac{1}{\lambda_\alpha(1-\lambda_\alpha)^2}$ (r_H)
1	1.0000	—	—	—
2	0.9867	$(0.9867)^3 \approx 0.9606$	$(0.9867)^4 \approx 0.9478$	$\frac{1}{0.9867(1-0.9867)^2} \approx 5722.41$
3	0.7000	$(0.7000)^3 \approx 0.3430$	$(0.7000)^4 \approx 0.2401$	$\frac{1}{0.7000(1-0.7000)^2} \approx 15.87$
4	0.4133	$(0.4133)^3 \approx 0.0706$	$(0.4133)^4 \approx 0.0292$	$\frac{1}{0.4133(1-0.4133)^2} \approx 7.02$

Table 4: Eigenvalues and spectral power factors ($t = 2$) for the 5000-vertex hypergraph.

5.6.5 Distance Computations: Full Spectrum vs. Single Dominant Eigenvalue (λ_2)

We evaluate five representative vertices across the topology to compare the full spectrum against the λ_2-truncated single-eigenvalue approximation:

- $v_1, v_1' \in C_1$: Internal core nodes of Cluster 1,
- $v_2 \in C_2$: Internal core node of Cluster 2,
- $v_3 \in C_3, v_4 \in C_4$: Internal core nodes of Clusters 3 and 4, respectively,
- $v_{34} \in C_3 \cap C_4$: A bridge node located on e_{bridge} belonging to both Cluster 3 and Cluster 4.

		Walktrap $r_V^{(2)}$		**Walktrap $r_{V\to\mathcal{C}}^{(2)}$**		**Hitting Time r_H**	
Pair	Structural Relation	Full	λ_2 only	Full	λ_2 only	Full	λ_2 only
(v_1, v_1')	Inside C_1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
(v_1, v_2)	$C_1 \leftrightarrow C_2$	0.4216	0.4132	0.4188	0.4105	**56.024**	**55.892**
(v_3, v_4)	$C_3 \leftrightarrow C_4$	0.3842	0.1850	0.3512	0.1837	4.218	1.240
(v_3, v_{34})	Core C3 $\leftrightarrow$ Bridge C3-C4	0.2014	0.0925	0.1820	0.0918	2.185	0.620
(v_1, v_3)	$C_1 \leftrightarrow C_3$ Cluster 3	0.5120	0.4850	0.5012	0.4818	**58.421**	**56.120**

Table 5: Comparison of pairwise distances ($t = 2$): Full Spectrum vs. Truncated (λ_2 only).

5.6.6 Comparative observations and partitioning granularity

From the calculations above, we observe that when λ_2 is close to 1, the commute hitting time distance is particularly effective because it significantly amplifies the inter cluster distances governed by this dominant eigenvalue. For instance, the distances $r_H(v_1, v_2)$ and $r_H(v_1, v_3)$ are substantially larger than $r_H(v_3, v_4)$, thereby cleanly partitioning the hypergraph into three macro components: C_1, C_2, and $C_3 \cup C_4$. In contrast, for the P^t based Walktrap distance, the scale variation between different community boundaries remains much less pronounced.

Furthermore, for hitting time, using solely λ_2 yields results nearly identical to those obtained from the full spectrum. Conversely, for P^t based distances, truncating the formulation to λ_2 alone leads to a significant deviation from the full-spectrum representation.

6 Community detection framework

6.1 Community detection via Walktrap

The Walktrap algorithm, originally proposed by Pons and Latapy [22], identifies community structures in networks by exploiting the properties of short random walks. This approach has also been investigated and extended in our previous works [6, 10, 9].

The algorithm is described as follows.

1. Initialization: The initial partition $\mathcal{P}_1 = \{\{v\} \mid v \in V\}$ consisting of $n = n$ communities, where each vertex forms its own community. The algorithm computes the Walktrap distances $d(u, v)$ between all adjacent vertices.

2. Iteration (step k), for $k = 1, 2, \ldots, n-1$:

   - Choose two adjacent communities in $\mathcal{P}_{k-1}$ that minimize the mean of the distances between each vertex and its community.
   - Merging these two communities into a new community, producing the updated partition.
   - Local distance update: computing the distances between the newly formed community and its neighboring adjacent communities.

3. Termination: After $n-1$ steps, the algorithm terminates at the trivial partition $\mathcal{P}_n = \{V\}$. The sequence of partitions $\{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_n\}$ forms a hierarchical tree structure called a

dendrogram, where the leaves represent individual vertices and internal vertices represent community merge operations.

The best partition is then considered to be the one that maximizes the mudularity Q_W.

Integration with spectral walk dynamics and weighted graph modularity

As proved in the previous section, the random walk transition dynamics over the hypergraph H (or equivalently, its vertex projection G_V and $G_{\mathcal{C}}$) are efficiently realized.

Moreover, the evaluating the modularity Q_W on G_V offers a distinct theoretical advantage. As analyzed in the preceding chapter, Q_W representations the sensitivity to community boundaries, effectively mitigating the classic resolution limit of unweighted graphs and enabling the detection of small scale community structures.

6.2 Embedding Space and PCA

In the previous section, we presented three distinct formulations for embedding the vertex set V into the r-dimensional Euclidean space $\mathbb{R}^r$, where $r \leq \rfloor \ll n$ denotes the rank of the incidence matrix B.

We summarize these dual spectral embedding frameworks as follows. Let $\widetilde{P}_{\mathcal{C}} = D_{\mathcal{C}}^{-\frac{1}{2}} B^T D_V^{-1} B D_{\mathcal{C}}^{-\frac{1}{2}}$ be the normalized hyperedge transition matrix, with positive non-zero eigenvalues $1 = \lambda_1 > \lambda_2 \geq \cdots \geq \lambda_r > 0$ and corresponding orthonormal hyperedge eigenvectors $s_{\mathcal{C},\alpha}$. Each vertex $v_i \in V$ is mapped into $\mathbb{R}^r$ under three distinct spectral representations:

1. **Vertex-to-Vertex Walktrap Embedding (f_V):** Maps each vertex $v_i \in V$ to an r-dimensional coordinate vector using a t-step random walk on the vertex space $V \to V$:

$$f_{V,\alpha}(v_i) = \frac{\lambda_\alpha^{t-\frac{1}{2}}}{d(v_i)} \left(B D_{\mathcal{C}}^{-\frac{1}{2}} s_{\mathcal{C},\alpha} \right)_i, \quad \text{for } \alpha = 1, \ldots, r.$$

2. **Cross-Set Walktrap Embedding ($f_{\mathcal{C}}$):** Maps each vertex $v_i \in V$ to an r-dimensional coordinate vector based on bipartite transitions between vertices and hyperedges $V \to \mathcal{C}$:

$$f_{\mathcal{C},\alpha}(v_i) = \frac{\lambda_\alpha^{t}}{d(v_i)} \left(B D_{\mathcal{C}}^{-\frac{1}{2}} s_{\mathcal{C},\alpha} \right)_i, \quad \text{for } \alpha = 1, \ldots, r.$$

3. **Commute Hitting Time Embedding (h_V):** Using the hitting time embedding, we define the map h_V which maps each vertex $v_i \in V$ to an n-dimensional profile vector $h_V(v_i) \in \mathbb{R}^n$:

$$h_V(v_i) = \left(\sqrt{\pi_1}\, h(v_i, v_1),\ \sqrt{\pi_2}\, h(v_i, v_2),\ \ldots,\ \sqrt{\pi_n}\, h(v_i, v_n) \right)^T. \tag{6.1}$$

Specifically, the component corresponding to target vertex v_k (where $k = 1, \ldots, n$) can be computed directly via the hyperedge spectrum as:

$$\left(h_V(v_i)\right)_k = m^{\frac{3}{2}} \sum_{\alpha \geq 2} \frac{1}{\lambda_\alpha (1 - \lambda_\alpha)} \left[\frac{\left(B D_{\mathcal{C}}^{-\frac{1}{2}} s_{\mathcal{C},\alpha} \right)_k}{(d(v_k))^{5/2}} - \frac{\left(B D_{\mathcal{C}}^{-\frac{1}{2}} s_{\mathcal{C},\alpha} \right)_i}{(d(v_k))^2 \sqrt{d(v_i)}} \right] \left(B D_{\mathcal{C}}^{-\frac{1}{2}} s_{\mathcal{C},\alpha} \right)_k. \tag{6.2}$$

Note that, although this embedding profile formula appears to be complicated, the structural distance $r_H(v_i, v_j) = \|h_V(v_i) - h_V(v_j)\|_2$ between any two embedded vertices simplifies significantly into an orthogonal spectral expansion:

$$r_H(v_i, v_j) = \sqrt{\sum_{\alpha \geq 2} \frac{m}{\lambda_\alpha (1-\lambda_\alpha)^2} \left(\frac{\left(BD_{\mathcal{C}}^{-\frac{1}{2}} s_{\mathcal{C},\alpha}\right)_i}{d(v_i)} - \frac{\left(BD_{\mathcal{C}}^{-\frac{1}{2}} s_{\mathcal{C},\alpha}\right)_j}{d(v_j)} \right)^2}. \tag{6.3}$$

6.3 Application to various community detection algorithm

The efficient distance computation scheme achieving $\mathcal{O}(nc)$ time complexity, combined with the **structural dimensionality reduction** from n dimensions to a compact k-dimensional space ($k \ll n$), provides a massive computational advantage for large-scale hypergraph analysis.

On one hand, projecting vertices into the k-dimensional Principal Component Space (PCS) allows spatial clustering algorithms like K**-means** to reduce their per-iteration time complexity from $\mathcal{O}(n^2)$ to merely $\mathcal{O}(n \cdot k)$, enabling extremely fast convergence. On the other hand, this low-dimensional representation enables seamless integration into modularity-based community detection heuristics such as the **Louvain** and **Leiden** algorithms. During local moving phases, evaluating the distance variation when reassigning a vertex across communities requires only $\mathcal{O}(k)$ time, allowing the overall framework to operate in near - linear time. Consequently, this framework scales naturally to real world hypergraphs comprising millions of vertices and hyperedges while retaining spectral accuracy.

7 From bipartite to tripartite and multilayer representations

The mathematical framework established for bipartite structures can be naturally generalized to a **tripartite representation** $T = (V, \mathcal{C}, K)$, where V denotes the set of vertices, $\mathcal{C}$ the set of groups (hyperedges), and K representing the interactions between groups.

Rather than a direct two step transition ($V \to \mathcal{C} \to V$), a random walk on T propagates through a four step cycle across the three layers:

$$V \xrightarrow{P_{V\to\mathcal{C}}} \mathcal{C} \xrightarrow{P_{\mathcal{C}\to\alpha}} K \xrightarrow{P_{K\to\mathcal{C}}} \mathcal{C} \xrightarrow{P_{\mathcal{C}\to V}} V.$$

7.1 Effective transition operators and random walk dynamics

Let $B \in \{0,1\}^{n\times\mathbf{c}}$ and $S \in \mathbb{R}_+^{\mathbf{c}\times\mathbf{k}}$ (where $\mathbf{k} = |K|$) denote the incidence matrices between adjacent layers. By composing the normalized projection matrices between consecutive layers, we define the induced Markov transition operator on V:

$$P_V = P_{\mathcal{C}\to V}\, P_{K\to\mathcal{C}}\, P_{\mathcal{C}\to K}\, P_{V\to\mathcal{C}} = D_V^{-1} B S S^\top B^\top,$$

where D_V is the diagonal degree matrix of V. The intermediate term $SS^\top$ acts as a nonlinear interaction kernel, modulating the effective connectivity between hyperedges.

From this induced operator P_V, all structural and diffusion properties originally designed for bipartite graphs can be systematically replicated and extended to T:

Diffusion istance & Walktrap: The t-step diffusion distance between two vertices $u, v \in V$ is defined via the t-step probability distribution $\Phi_t(u) = P_V^t(u, \cdot)$:

$$d_V^{(t)}(u,v)^2 = \|\Phi_t(u) - \Phi_t(v)\|^2_{D_V^{-1}} = \sum_{x \in V} \frac{\left((P_V^t)_{ux} - (P_V^t)_{vx}\right)^2}{d(x)}.$$

To mitigate the computational cost of exponentiating P_V when n is large, we reduce the calculation to the group-level chain $P_{\mathcal{C}} = P_{\mathcal{C}\to K} P_{K\to\mathcal{C}}$, yielding $P_V^t = P_{V\to\mathcal{C}}\, P_{\mathcal{C}}^{t-1}\, P_{\mathcal{C}\to V}$.

Effective Weighted Adjacency & Modularity: The tripartite interaction induces an effective edge weight $w_V(u, v)$ between vertices:

$$w_V(u,v) = \sum_{c,c' \in \mathcal{C}} \sum_{k \in K} \frac{B_{u,c}\, S_{c,k}\, S_{c',k}\, B_{v,c'}}{d(c) d(c')}.$$

Let $W_V = (w_V(u, v))$ be the resulting adjacency matrix, $d_V(u) = \sum_v w_V(u, v)$, and $2m_V = \sum_{u,v} w_V(u, v)$. The Newman-Girvan modularity $Q(\mathcal{A})$ for a partition $\mathcal{A}$ on V remains structurally consistent:

$$Q(\mathcal{A}) = \frac{1}{2m_V} \sum_{u,v \in V} \left(w_V(u,v) - \frac{d_V(u) d_V(v)}{2m_V} \right) \mathbf{1}_{\{u,v \text{ in the same cluster}\}}.$$

Stationary Distribution & Spectral Inheritance: The stationary distribution π_V on V satisfies $\pi_V P_V = \pi_V$. Moreover, the low lying spectrum and spectral gap γ_α of the operator $P_K = P_{\mathcal{C}\to K} P_{K\to\mathcal{C}}$ propagate through the intermediate layers, directly governing the long term diffusion dynamics and community bottlenecks on V.

7.2 Generalization to arbitrary multilayer structures

In network science, researchers often employ multigraphs to capture multiple levels of interaction among vertices [3, 4, 13]. Alternatively, if one is interested in the set of maximal cliques, the network can be modeled as a bipartite graph where the first level represents the vertices and the second level represents the maximal cliques. Furthermore, to account for intersections between cliques, a third level representing these intersections can be added, naturally extending the model to a multilevel hierarchy. This framework demonstrates that introducing intermediate layers enriches the transition kernel without altering the underlying theoretical methodology. Consequently, this formulation generalizes to any arbitrary multilayer network consisting of ℓ consecutive layers $(L_1, L_2, \ldots, L_\ell)$:

1. Inter layer projections: Define normalized projection operators $P_{i\to i+1}$ between adjacent layers L_i and L_{i+1} for $i = 1, \ldots, L-1$.

2. Induced effective kernel: The forward and backward propagation through all layers defines a unified Markov operator on the target layer L_1:

$$P_{L_1} = P_{L_2\to L_1}\, P_{L_3\to L_2} \ldots P_{L_\ell\to L_{\ell-1}}\, P_{L_{\ell-1}\to L_\ell} \ldots P_{L_1\to L_2}.$$

3. Unified Community Detection: Once P_{L_1} or its corresponding weighted matrix W_{L_1} is established, standard graph-based algorithms, such as Walktrap, Louvain, or spectral clustering, can be applied directly to discover higher order community structures across complex multilayer systems.

7.3 Generalization to arbitrary bipartite networks.

While our framework originates from hypergraph analysis, where a hypergraph H is mapped onto a bipartite representation $G_B = (V \cup \mathcal{C}, E_B)$ such that a standard random walk on G_B translates into a weighted random walk on the projected vertex graph G_V. Specifically, this reduction does not require the set $\mathcal{C}$ to represent hyperedges or clusters; it applies directly to arbitrary bipartite structures where nodes in V and $\mathcal{C}$ hold symmetric structural roles. Under this unified perspective, a unweighted random walk on G_B can be decomposed into two coupled weighted random walks operating on G_V and $G_{\mathcal{C}}$ over even and odd time steps, respectively. Consequently, fundamental topological quantities on G_B, including stationarity, modularity, spectral distances, and hitting times, can be efficiently evaluated directly through the lower dimensional transition operators P_V and $P_{\mathcal{C}}$. This duality provides a versatile computational paradigm for distance estimation, community detection, and dimensionality reduction on general bipartite networks.

8 Conclusion

In this paper, we have constructed a framework for studying complex systems in which elements are connected within groups, modeled by a hypergraph $H = (V, \mathcal{C})$. From this hypergraph, we construct an equivalent bipartite graph $G_B = (V \cup \mathcal{C}, E_B)$ and construct projections onto the vertex graph $G_V = (V, E_V)$ and the hyperedge graph $G_{\mathcal{C}} = (\mathcal{C}, E_{\mathcal{C}})$. We also construct a random walk on G_B that naturally induces random walks with transition probabilities corresponding to the edge weights on G_V and $G_{\mathcal{C}}$. Because both random walks are induced from the same random walk on G_B, their corresponding transition probability matrices share the same spectrum (they have the same r positive eigenvalues and all other eigenvalues are zero). Consequently, spectral calculations on G_V and $G_{\mathcal{C}}$ can be derived from one another. In many practical cases where $\mathbf{c} \ll n$, computations can be performed on $G_{\mathcal{C}}$ instead of G_V, thereby significantly reducing the computational dimensionality and lowering the complexity of spectral decomposition and matrix multiplication operations from size n to $\mathbf{c}$.

Our proposed framework serves as a theoretical foundation for graph embedding techniques into vector spaces $\mathbb{R}^n$, $\mathbb{R}^{\mathbf{c}}$, or $\mathbb{R}^r$. Based on this, established embedding methods can be implemented, such as those based on t-step transition probabilities $P^t_{v,\cdot}$, hitting times $h(v, \cdot)$, or matrix spectra. In particular, the computational is substantially reduced due to the above spectral equivalence property, giving the development of efficient spectral and PCA based algorithms.

An important characteristic of this approach is that the modularity calculated on this weighted projection graph is superior to the modularity on unweighted projection graphs, as it effectively captures small size communities and facilitates the resolution and separation of both micro and macro communities. Consequently, classic community detection algorithms such as Louvain, Leiden, and Newman can achieve enhanced performance.

In this work, rather than investigating extensive empirical experiments, we establish the theoretical foundation accompanied by an analysis of basic structural motifs that may appear in hypergraphs. Substantial methods for future research remain open when applying this framework to synthetic benchmark datasets or real world networks, as well as extending it to large scale network algorithms, node similarity measures, structural property discovery, and downstream community detection tasks.

Finally, our work outlines a natural extension to multilayer graphs beyond two levels. For

example, a three level graph structure $G = (V, \mathcal{C}, K)$ can naturally model academic domain classifications, where V represents authors, $\mathcal{C}$ represents articles, and K represents research subjects studied across citation networks. This offers a promising direction for future work, particularly by conducting in depth analyzes of random walks between adjacent layers to formally define novel graph embeddings and multilevel community partitioning techniques.

References

[1] 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:P10008, 2008.

[2] Uthsav Chitra and Benjamin J Raphael. Random walks on hypergraphs with edge-dependent vertex weights. In *International Conference on Machine Learning (ICML)*, pages 1172–1181. PMLR, 2019.

[3] C. Christophe, L. Matthieu, and T. H. D. Phan. On the termination of some biclique operators on multipartite graphs. *Discrete Appl. Math.*, 195:59–73, 2015.

[4] C. Christophe, T.H.D. Phan, and T. H. Tran. Termination of the iterated strong-factor operator on multipartite graphs. *Theoret. Comput. Sci.*, 571:67–77, 2015.

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

[6] Tien Dat Dang, Duy Hieu Do, and Thi Ha Duong Phan. Community detection in directed graphs using stationary distribution and hitting times methods. *Social Network Analysis and Mining*, 13(80), 2023.

[7] Tien Dat Dang, Duy Hieu Do, and Thi Ha Duong Phan. Community detection in directed graphs using stationary distribution and hitting times methods. *Social Network Analysis and Mining*, 13:1–30, 2023.

[8] D. T. Dat, D. D. Hieu, and P. T. H. Duong. Community detection in directed graphs using stationary distribution and hitting times methods. *Social Network Analysis and Mining*, 13(80), 2023.

[9] Duy Hieu Do and Phan Thi Ha Duong. An improvement on the louvain algorithm using random walks. *Journal of Combinatorial Optimization*, 50(14), 2025.

[10] Duy Hieu Do and Thi Ha Duong Phan. Overlapping community detection algorithms using modularity and the cosine. *Advances in Complex Systems*, 28(03):2550006, 2025.

[11] Anton Eriksson, Daniel Edler, Alexis Rojas, Manlio De Domenico, and Martin Rosvall. Choosing random-walk model and network representation matters for flow-based community detection in hypergraphs. *Communications Physics*, 4(1):133, 2021.

[12] Gary W. Flake, Steve Lawrence, C. Lee Giles, and Frans M. Coetzee. Self-organization and identification of web communities. *Computer*, 35(3):66–71, 2002.

[13] Jean-Loup Guillaume and Matthieu Latapy. Bipartite graphs as models of complex networks. *Physica A: Statistical Mechanics and its Applications*, 371(2):795–813, 2006.

[14] Koby Hayashi, Sinan G Aksoy, Cheong Hee Park, and Haesun Park. Hypergraph random walks, laplacians, and clustering. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management (CIKM)*, pages 495–504, 2020.

[15] Jon Kleinberg and Steve Lawrence. The structure of the web. *Science*, 294(5548):1849–1850, 2001.

[16] E. A. Leicht and M. E. J. Newman. Community structure in directed networks. *Phys. Rev. Lett.*, 100:118703, Mar 2008.

[17] Cristopher Moore. The computer science and physics of community detection: Landscapes, phase transitions, and hardness. *Bulletin of the EATCS*, 121, 2017.

[18] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[19] M. E. J. Newman. Spectral methods for network community detection and graph partitioning. *Physical Review E*, 88:042822, 2013.

[20] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2):026113, 2004.

[21] Mark Newman, Mark Newman, Michelle Girvan, and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 69 2 Pt 2:026113, 2003.

[22] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. In *J. Graph Algorithms Appl.*, 2004.

[23] Ehab Ramadan, Ashis Tarafdar, and Alex Pothen. A hypergraph model for the yeast protein complex network. In *18th International Parallel and Distributed Processing Symposium, 2004. Proceedings.*, page 189. IEEE, 2004.

[24] Erzsébet Ravasz, Alexander L. Somera, D. A. Mongru, Zoltán N. Oltvai, and Albert-László Barabási. Hierarchical organization of modularity in metabolic networks. *Science*, 297(5586):1551–1555, 2002.

[25] Anna Ritz, Allison N Tegge, Hyunju Kim, Christopher L Poirel, and TM Murali. Signaling hypergraphs. *Trends in Biotechnology*, 32(7):356–362, 2014.

[26] Christiane Takács. On the fundamental matrix of finite state markov chains, its eigensystem and its relation to hitting times. *Mathematica Pannonica*, 17(2):183–193, 2006.

[27] Vincent A. Traag, Ludo Waltman, and Nees Jan van Eck. From louvain to leiden: guaranteeing well-connected communities. *Scientific Reports*, 9:5233, 2019.

[28] Ludo Waltman and Nees Jan van Eck. A smart local moving algorithm for large-scale modularity-based community detection. *The European Physical Journal B*, 86:471, 2013.

[29] Yan Wang, Jie Liang, and Jianguo Lu. Discover hidden web properties by random walk on bipartite graph. *Information Retrieval*, 17(3):203–228, 2014.

[30] Yanbang Wang and Jon Kleinberg. From graphs to hypergraphs: Hypergraph projection and its reconstruction. In *The Twelfth International Conference on Learning Representations*, 2024.

[31] Wenyin Yang, Guojun Wang, Md Zakirul Alam Bhuiyan, and Kim-Kwang Raymond Choo. Hypergraph partitioning for social networks based on information entropy modularity. *Journal of Network and Computer Applications*, 86:59–71, 2017.

[32] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. In *Advances in Neural Information Processing Systems (NIPS)*, volume 19, pages 1601–1608, 2006.