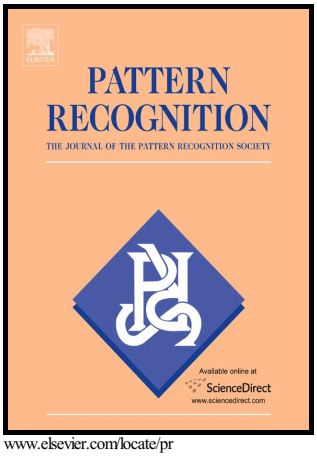
Depth-based Hypergraph Complexity Traces from Directed Line Graphs

Lu Bai, Francisco Escolano, Edwin R. Hancock



www.elsevier.com/locate/pr

PII: S0031-3203(16)00006-6
DOI: http://dx.doi.org/10.1016/j.patcog.2016.01.004
Reference: PR5604

To appear in: *Pattern Recognition*

Received date: 15 November 2015
Accepted date: 7 January 2016

Cite this article as: Lu Bai, Francisco Escolano and Edwin R. Hancock, Depth based Hypergraph Complexity Traces from Directed Line Graphs, *Patter Recognition,* http://dx.doi.org/10.1016/j.patcog.2016.01.004

Depth-based Hypergraph Complexity Traces from Directed Line Graphs

Lu Bai[1*], Francisco Escolano [2**], Edwin R. Hancock[3]

[1] *School of Information, Central University of Finance and Economics, Beijing, China.*
[2] *Department of Computer Science and Artificial Intelligence, University of Alicante, Spain.*
[3] *the Department of Computer Science, University of York, York, UK.*

Abstract

In this paper, we aim to characterize the structure of hypergraphs in terms of structural complexity measure. Measuring the complexity of a hypergraph in a straightforward way tends to be elusive since the hyperedges of a hypergraph may exhibit varying relational orders. We thus transform a hypergraph into a line graph which not only accurately reflects the multiple relationships exhibited by the hyperedges but is also easier to manipulate for complexity analysis. To locate the dominant substructure within a line graph, we identify a centroid vertex by computing the minimum variance of its shortest path lengths. A family of centroid expansion subgraphs of the line graph is then derived from the centroid vertex. We compute the depth-based complexity traces for the hypergraph by measuring either the directed or undirected entropies of its centroid expansion subgraphs. The resulting complexity traces provide a flexible framework that can be applied to both hypergraphs and graphs. We perform (hyper)graph classification in the principal component space of the complexity trace vectors. Experiments on (hyper)graph datasets abstracted from bioinformatics and computer vision data demonstrate the effectiveness and efficiency of the complexity traces.

Keywords: Hypergraphs, Directed Line Graphs, Entropies, Centroid Vertex, Depth-based Complexity Traces

*Email address: bailucs@cufe.edu.cn and bailu69@hotmail.com.
**Email address: escolano.ua@gmail.com.
Email address: erh@cs.york.ac.uk.

1. Introduction

There has recently been an increasing interest in the use of hypergraph models for higher order learning. A hypergraph is a generalization of a graph. Unlike the pairwise nature of edges in a graph, hypergraph representations allow a hyperedge to encompass an arbitrary number of vertices, and can hence capture multiple relationships among features.

Most existing methods attempt to approximate a hypergraph by an equivalent graph, and exploit existing graph based methods for learning higher order models. For instance, Agarwal et al. [1] have performed hypergraph clustering by partitioning a weighted graph obtained by transforming the original hypergraph using a weighted sum of hyperedges to form edges. Zhou et al. [2], on the other hand, have presented a similar graph approximation method for hypergraphs by normalizing the Laplacian matrix of the star expansion of a hypergraph. Wachman et al. [3] have developed a hypergraph kernel by enumerating similar walks on two hypergraphs. Zass et al. [4] and Duchenne et al. [5] have separately applied high-degree affinity arrays (i.e. tensors) to formulate hypergraph matching problems using different cost functions. Both methods address the matching process in an algebraic manner but become intractable to compute if the hyperedges are not suitably sampled. Shashua et al. [6, 7] have performed visual clustering using tensors to represent uniform hypergraphs (i.e. those for which the hyperedges have identical cardinality) extracted from images and videos. Their work has been complemented by He et al.'s [8] algorithm for detecting number of clusters in a tensor-based framework. Similar methods include those described in [9, 10, 11, 12, 13], in which tensors (uniform hypergraphs) are used to represent the multiple relationships between objects.

One limitation of most existing methods for hypergraph characterization is that they are usually restricted to uniform structures and cannot be applied to hypergraphs with arbitrary relational orders. To address this shortcoming, Ren et al. [14] have exploited a set of polynomial coefficients obtained from the hypergraph Ihara zeta function for characterizing nonuniform hypergraphs. Unfortunately, the computation of the hypergraph Ihara coefficients tends to be computational burdensome.

The aim of this paper is to overcome the limitations of existing methods for hypergraph anal-

ysis by computing a depth-based complexity trace for a hypergraph. We have previously explored this idea for ordinary graphs [32], but we have not shown how to extend the idea to hypergraphs. Our idea is to transform a hypergraph into an equivalent directed line graph that accurately captures the multiple relationship exhibited by the hypergraph. The complexity trace of a hypergraph can thus be computed by measuring the information content of a family of expansion subgraphs that are derived from the centroid vertex of its line graph. Specifically, we explore how to characterize the layer expansion subgraphs using complexity measures as a function of depth.

1.1. Literature Review

Computing the entropy-based complexity of an undirected (sub)graph has attracted significant attention due to its fundamental practical importance for network analysis [32]. In early work, Körner [16] developed a classical undirected graph entropy which poses complexity characterization as an optimization problem. This approach is based on a probability distribution associated with the vertices, and the complexity is the minimal cross entropy between the probability distribution and the vertex packing polytype of the graph. Unfortunately, this entropy is not applicable to more general unweigthed graphs. Mowshowitz [17], Rashevsky [18] and Trucco [19] have each developed a Shannon entropy for a graph associated with the probability distribution derived from different partitionings of the vertex set. Unfortunately, determining the partitioning requires expansive computation, thus the entropy cannot be efficiently computed. To overcome the shortcomings for these classical graph entropies, Dehmer et al. [20, 21] have developed a novel means of computing entropies of undirected graphs by using information functionals. The information functionals for an undirected graph are derived from the topological structure of the graph and quantify the information content of the given graph structure. Moreover, this approach avoids the combinatorial computations over different vertex partitions by constructing local information subgraphs for a given undirected graph, and thus achieves a polynomial time complexity. Anand et al. [23] and Passerini et al. [22] have applied the von Neumann entropy (or quantum entropy) to the domain of graphs through a mapping between discrete Laplacians and quantum states [24]. If the discrete Laplacian [25] is scaled by the inverse of the volume of the graph we obtain a density matrix whose entropy can be computed using the spectrum of the discrete Laplacian. The mea-

sure distinguishes between different structures. For instance it is maximal for random undirected graphs, minimal for complete ones and takes on intermediate values for star graphs. In addition, when there is degree heterogeneity then the Shannon (classical) and von Neumann (quantum information theoretic) entropies are correlated. However, since the von Neumann entropy relies on the computation of the normalized Laplacian spectrum, its computational complexity is cubic in the number of vertices.

To render the computation of the von Neumann entropy more efficient, Han et al. [26] have shown how the computation can be rendered quadratic in the number of the vertices by approximating the Shannon entropy on the Laplacian eigenvalues using its quadratic counterpart. An analysis of the quadratic entropy reveals that it can be computed from a number of permutation invariant matrix trace expressions. This leads to a simple expression for the approximate entropy in terms of the degrees of the adjacent vertices. Furthermore, to develop this work further, Ye et al. [28] have shown how the von Neumann entropy for undirected graphs can be extended to directed graphs. To do this, they commenced by using Chung's [29] definition of the normalized Laplacian on a directed graph. According to this definition, the directed graph normalized Laplacian is Hermitian, and so the interpretation of Passerini et al. in [23] still holds for the domain of directed graphs. The von Neumann entropy is essentially the Shannon entropy associated with the normalized Laplacian eigenvalues. The resulting von Neumann entropy expression for directed line graphs depends on the in-degree and out-degree of pairs of vertices connected by edges.

Recently, depth-based representations of undirected graph structures have been widely used for developing new complexity measures for graphs [30, 31]. One approach is to gauge information content flow through K-layer subgraphs of a graph (e.g. subgraphs around a vertex having a maximum topological distance or minimal path length K) of increasing size and to use the flow as a structural signature. Following this approach, Bai and Hancock [32, 33, 34] have developed a fast depth-based complexity trace from the centroid vertex that has the minimum variance of shortest path lengths to the remaining vertices (i.e., a depth-based representation around the centroid vertex). They decompose a graph into a family of K-layer centroid expansion subgraphs around the centroid vertex. A complexity trace vector is computed by measuring the entropies on the individual expansion subgraphs. Since the centroid based method can be used to efficiently

compute the entropy based complexity measures on a small set of expansion subgraphs rooted at the centroid vertex, it can be thus computed in a polynomial time. By contrast, the thermodynamic depth complexity measure developed in [30] requires the expansion subgraphs rooted at each vertex, and computes the intrinsic complexity for each subgraph. It is thus less efficient to compute on large graphs, e.g., a graph having thousands of vertices.

Unfortunately, all of the above mentioned complexity measures, both entropy-based and depth-based, are only developed for (un)directed graphs and do not easily accommodate hypergraphs. The reason for this is that straightforwardly measuring the complexity of a hypergraph tends to be an elusive problem since the hyperedge in a hypergraph may exhibit varying relational orders. Therefore, to measure the complexity of a hypergraph, in a manner that can precisely capture the structural information contained within it, we consider transforming a hypergraph into a directed line graph using the Perron-Frobenius operator [14]. The Perron-Frobenius operator can represent both uniform or nonuniform hypergraphs characteristics and can also accurately reflect the multiple relationships exhibited by hyperedges of different orders. Hence, the directed line graph representation for a hypergraph provides a convenient framework for complexity analysis.

1.2. Contributions

As previously stated, the aim of this paper is to present a novel framework for characterizing hypergraphs based on computing depth-based complexity traces. Our starting point is the line graph obtained by transforming a hypergraph into substructures using the Perron-Frobenius operator. The complexity of the hypergraph is then measured by computing the entropies of the substructures. We develop two different classes of complexity traces for a hypergraph HG based on two different decomposition strategies. The first is to establish an undirected line graph G_U of HG from the equivalent directed line graph G_{DL} by simply neglecting the edge directions of G_{DL}. We derive a family of expansion subgraphs with increasing layer size K from G_U. We construct an undirected complexity trace of HG by measuring how the undirected entropy measure varies on the expansion subgraphs with increasing K. The second strategy is to establish a directed complexity trace for HG by measuring the directed graph entropy on a family of expansion subgraphs derived from the directed line graph G_{DL} of HG. Since we measure the entropy over the family of

expansion subgraphs, both methods are efficient and overcome the computational bottlenecks existing in state-of-the-art methods for network complexity analysis [26, 30]. Our hypergraph complexity traces provide a flexible framework that can be applied to both hypergraphs and graphs. We perform experiments on several bioinformatics and computer vision datasets. We empirically demonstrate that our complexity traces not only readily accommodate nonuniform hypergraphs, but also easily scale to large hypergraphs. The performance of our framework is competitive with alternative network complexity analysis methods and other hypergraph based methods reported in the literature.

The remainder of this paper is organized as follows: Section 2 and Section 3 respectively introduce the entropy measures for undirected or directed graphs that will be used in our framework. For an undirected graph, Section 4 presents a family of centroid expansion subgraph that will also be used in our framework. Section 5 describes how to compute an undirected or a directed depth-based complexity trace for a hypergraph. Section 6 provides experimental comparisons between the proposed hypergraph complexity trace methods and state-of-the-art (hyper)graph based methods. Section 7 concludes this work and makes suggestions for future work.

2. Entropy Measures on Undirected Graphs

In this section, we review how to compute the entropy for an undirected graph. We commence by reviewing the concept of von Neumann entropy used in previous work [26]. Here we commence by explaining how the von Neumann entropy of an undirected graph can be efficiently computed in terms of the degree statistics using a quadratic approximation to the Shannon entropy. We also introduce an alternative Shannon entropy using the probability distribution associated with a steady state random walk on an undirected graph [32, 35].

2.1. Von Neumann Entropy of Undirected Graphs

The von Neumann entropy of an undirected graph is the Shannon entropy associated with the eigenvalues of the normalized undirected graph Laplacian [22]. We denote the undirected graph under study by $G(V, E)$ where V is the set of vertices and $E \subseteq V \times V$ is the set of undirected

edges. The symmetric adjacency matrix A for $G(V,E)$ is a $|V| \times |V|$ matrix that has elements

$$A(i,j) = \begin{cases} 1 & \text{if}(v_i, v_j) \in E; \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

The vertex degree matrix of $G(V,E)$ is a diagonal matrix D whose elements are given by $D(v_i,v_i) = d(i) = \sum_{v_j \in V} A(i,j)$. From the degree matrix and the adjacency matrix we can construct the Laplacian matrix $L = D - A$. The normalized Laplacian matrix is given by $\hat{L} = D^{-1/2}LD^{-1/2}$. The spectral decomposition of the normalized Laplacian matrix is $\hat{L} = \hat{\Phi}\hat{\Lambda}\hat{\Phi}^T$ where $\hat{\Lambda} = diag(\hat{\lambda}_1, \hat{\lambda}_2, ..., \hat{\lambda}_{|V|})$ is a diagonal matrix with the ordered eigenvalues as elements $(0 = \hat{\lambda}_1 < \hat{\lambda}_2 < ... < \hat{\lambda}_{|V|})$ and $\hat{\Phi} = (\hat{\phi}_1|\hat{\phi}_2|...|\hat{\phi}_{|V|})$ is a matrix with the corresponding ordered orthonormal eigenvectors as columns. The normalized Laplacian matrix is positive semi-definite and so has all eigenvalues non-negative. The number of zero eigenvalues is the number of connected components in $G(V,E)$. The von Neumann entropy of $G(V,E)$ associated with the normalized Laplacian eigenspectrum [22] is defined as

$$H_{VN} = -\sum_{i=1}^{|V|} \frac{\hat{\lambda}_i}{|V|} \log \frac{\hat{\lambda}_i}{|V|} \tag{2}$$

The computation of the von Neumann entropy requires a number of operations that is cubic in the number of vertices $|V|$, since it requires the solution of the eigendecomposition. Han et al. [26] have shown how the computation can be computed in quadratic time by a) approximating the Shannon entropy by its quadratic counterpart, and b) evaluating the traces of $\hat{L}$ and $\hat{L}^2$ using vertex degrees. To commence, they approximate the Shannon entropy $\frac{\hat{\lambda}_i}{|V|} \ln \frac{\hat{\lambda}_i}{|V|}$ by its quadratic counterpart $\frac{\hat{\lambda}_i}{|V|}(1 - \frac{\hat{\lambda}_i}{|V|})$ and obtain

$$\begin{aligned} H_{VN} &= -\sum_{i=1}^{|V|} \frac{\hat{\lambda}_i}{|V|} \log \frac{\hat{\lambda}_i}{|V|} \simeq \sum_{i=1}^{|V|} \frac{\hat{\lambda}_i}{|V|}(1 - \frac{\hat{\lambda}_i}{|V|}) \\ &= \frac{1}{|V|}\sum_{i=1}^{|V|} \hat{\lambda}_i - \frac{1}{|V|^2}\sum_{i=1}^{|V|} \hat{\lambda}_i^2. \end{aligned} \tag{3}$$

Based on the definition by Han et al. in [26], $\sum_{i=1}^{|V|} \hat{\lambda}_i = Tr[\hat{L}] = |V|$, and $\sum_{i=1}^{|V|} \hat{\lambda}_i^2 = Tr[\hat{L}^2] = |V| + \sum_{(v_i,v_j)\in E} \frac{1}{d(i)d(j)}$. Thus, the von Neumann entropy defined in Eq.(3) can be re-written as

$$H_{VN}(G) = 1 - \frac{1}{|V|} - \frac{1}{|V|^2} \sum_{(v_i,v_j)\in E} \frac{1}{d(i)d(j)} \tag{4}$$

As a result, we can approximate the von Neumann entropy using two measures of an undirected graph structure. The first is the number of vertices, and the second is based on degree statistics for pairs of vertices connected by edges. The approximation bypasses calculating the normalized Laplacian eigenvalues of an undirected graph which is ($O(|V|^3)$). Therefore, we estimate the von Neumann entropy in time $O(|V|^2)$, and this renders the computation more efficient.

2.2. Shannon Entropy of Undirected Graphs

An alternative approach to computing the entropy of $G(V, E)$ is to use a steady state random walk on $G(V, E)$. The probability of the steady state random walk on $G(V, E)$ visiting vertex v_i is

$$P(i) = d(i)/\sum_{v_j \in \mathcal{V}} d(j) \ . \tag{5}$$

Based on Eq.(5), we obtain a probability distribution P associated with the steady state random walk on $G(V, E)$, and the Shannon entropy for $G(V, E)$ is given by

$$H_S(G) = -\sum_{i=1}^{|V|} P(i) \log P(i). \tag{6}$$

For the undirected graph $G(V, E)$, computing the Shannon entropy $H_S(G)$ requires $O(|V|^2)$ operations, because it needs to visit all the $|V|^2$ pairs of entries in A to compute the probability of a steady state random walk visiting each vertex. This indicates that the Shannon entropy associated with a steady state random walk can be efficiently computed.

3. Entropy Measures on Directed Graphs

The entropy measures defined in Section 2 only apply to graphs with undirected edges. However, in our study, we also require entropy measures on graphs with directed edges (see Section 5.3 for details). In this section, we introduce two entropy measures for directed graphs. We commence by introducing a directed von Neumann entropy [28]. This method is based on extending the definition of the von Neumann entropy (i.e., the von Neumann entropy of undirected graphs defined in Section 2) from undirected to directed graphs, and is expressed in terms of the in-degree and out-degree statistics of vertices. Moreover, we also introduce an asymptotic entropy in terms of the heat flow diffusion [30].

3.1. Von Neumann Entropy of Directed Graphs

Let $G_D(V_D, E_D)$ is a directed graph with vertex set V_D and edge set $E_D \subseteq V_D \times V_D$, and A_D is the adjacency matrix of G_D. The in-degree and out-degree of vertex $v_{D;i}$ are

$$d_{in}(i) = \sum_{j=1}^{|V_D|} A_D(j,i), \quad d_{out}(j) = \sum_{j=1}^{|V_D|} A_D(i,j). \tag{7}$$

With these ingredients, the transition matrix $\mathcal{T}$ for the directed graph G_D is defined as

$$\mathcal{T}(i,j) = \begin{cases} \frac{A_D(i,j)}{d_{out}(i)} & \text{if } (v_{D;i}, v_{D;j}) \in E_D \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

In [29], Cheng has shown that the normalized Laplacian matrix of a directed graph can be written as

$$\tilde{L}_D = I - \frac{\Phi_D^{1/2}\mathcal{T}\Phi_D^{-1/2} + \Phi_D^{-1/2}\mathcal{T}^T\Phi_D^{1/2}}{2}, \tag{9}$$

where $\Phi_D = diag(\phi_{D;1}, \phi_{D;2}, \ldots, \phi_{D;|V_D|})$ and $\phi_D = (\phi_{D;1}|\phi_{D;2}|\ldots|\phi_{D;|V_D|})$ is the left eigenvector of $\tilde{L}_D$.

Similar to the von Neumann entropy of an undirected graph [22], the von Neumann entropy for a directed graph can also be approximated using the Shannon entropy associated with the eigenvalues of its normalized Laplacian matrix. Using the approximation, in [28] Ye at al. have extended the analysis of Han et al. [26] from undirected to directed graphs. The starting point is the quadratic approximation to the von Neumann entropy in terms of the traces of normalized Laplacian and the squared normalized Laplacian, i.e.,

$$H_{VN}^D = \frac{Tr[\tilde{L}_D]}{|V_D|} - \frac{Tr[\tilde{L}_D^2]}{|V_D|^2}. \tag{10}$$

To simplify Eq.(10) one step further, let $E_{D;1}$ and $E_{D;2}$ are two disjoint subsets of E_D, and satisfy $E_{D;1} = \{(v_{D;i}, v_{D;j})|(v_{D;i}, v_{D;j}) \in E_D \wedge (v_{D;j}, v_{D;i}) \notin E_D\}$ and $E_{D;2} = \{(v_{D;i}, v_{D;j})|(v_{D;i}, v_{D;j}) \in E_D \wedge (v_{D;j}, v_{D;i}) \in E_D\}$. $E_{D;1} \bigcup E_{D;2} = E_D$, and $E_{D;1} \bigcap E_{D;2} = \emptyset$. Based on [28], we have

$$Tr[\tilde{L}_D] = Tr[I] = |V_D|,$$

and

$$Tr[\tilde{L}_D^2] = |V| + \frac{1}{2}(Tr[\mathcal{T}^2] + Tr[\mathcal{T}\Phi_D^{-1}\mathcal{T}^T\Phi_D]),$$

where $Tr[\mathcal{T}^2] = \sum_{(v_i,v_j)\in E_{D;2}} \frac{1}{d_{out}(i)d_{out}(j)}$ and $Tr[\mathcal{T}\Phi^{-1}\mathcal{T}^T\Phi] = \sum_{(v_i,v_j)\in E_D} \frac{\phi(i)}{\phi(i)d_{out}(j)^2}$. Using the fact that $\frac{\phi_{D;i}}{\phi_{D;j}} \approx \frac{d_{in}(i)}{d_{in}(j)}$ [28], we can approximate the von Neumann entropy of a directed graph in terms of the in-degree and out-degree of the vertices as follows

$$H_{VN}^D = 1 - \frac{1}{|V_D|} - \frac{1}{2|V_D|^2}\Bigg\{\sum_{(v_i,v_j)\in E_D}\Bigg(\frac{1}{d_{out}(j)d_{out}(i)} + \frac{d_{in}(j)}{d_{in}(i)d_{out}(j)^2}\Bigg) - \sum_{(v_i,v_j)\in E_{D;1}} \frac{1}{d_{out}(j)d_{out}(i)}\Bigg\}, \tag{11}$$

or equivalently,

$$H_{VN}^D = 1 - \frac{1}{|V_D|} - \frac{1}{2|V_D|^2}\Bigg\{\sum_{(v_{D;i},v_{D;j})\in E_D} \frac{d_{in}(i)}{d_{in}(j)d_{out}(i)^2} + \sum_{(v_{D;i},v_{D;j})\in E_{D;2}} \frac{1}{d_{out}(i)d_{out}(j)}\Bigg\}. \tag{12}$$

Eq.(11) or Eq.(12) can be consequently simplified according to the relative sizes of the sets $E_{D;1}$ and $E_{D;2}$. If G_D is a weakly directed graph ($|E_{D;1}| \ll |E_{D;2}|$), i.e., few of the edges are not bidirectional, the approximate von Neumann entropy is defined as [28]

$$H_{VN}^{WD} = 1 - \frac{1}{|V_D|} - \frac{1}{2|V_D|^2} \sum_{(v_{D;i},v_{D;j})\in E_D} \frac{\frac{d_{in}(i)}{d_{out}(i)} + \frac{d_{in}(j)}{d_{out}(j)}}{d_{out}(i)d_{in}(j)}. \tag{13}$$

On the other hand, if G_D is a strongly directed graph ($|E_{D;2}| \ll |E_{D;1}|$), i.e., there are few bidirectional edges, the approximate von Neumann entropy is given by [28]

$$H_{VN}^{SD} = 1 - \frac{1}{|V_D|} - \frac{1}{2|V_D|^2} \sum_{(v_{D;i},v_{D;j})\in E_D} \Bigg\{\frac{1}{d_{out}(i)d_{in}(j)}\Bigg\}. \tag{14}$$

Both the weakly and strongly directed forms of the von Neumann entropy (H_{VN}^{WD} and H_{VN}^{SD}) contain two terms. The first is the graph size while the second one depends on the in-degree and out-degree statistics of each pair of vertices linked by an edge. Moreover, the computational complexity of these expressions is quadratic in the graph size.

3.2. Asymptotic (Flow) Entropy of Directed Graphs

Heat kernels are the solution $K(\beta)$ to the heat/diffusion equation: $\frac{\partial K(\beta)}{\partial \beta} = -LK(\beta)$, where β means time [37] and L is the graph Laplacian. Diffusion kernels are doubly stochastic matrices.

Since $K(\beta)$ is semi-definite positive we have that the spectral decomposition $K(\beta) = \Psi e^{-\beta\Lambda}\Psi^T$ where $\Lambda = diag(\lambda_1, \ldots, \lambda_{|V|})$ contains the eigenvalues $0 = \lambda_1 < \lambda_2 < \ldots < \lambda_{|V|}$ and $\Psi = (\psi_1|\psi_2|\ldots|\psi_{|V|})$ the eigenvectors, leads to $\lim_{\beta\to\infty} K(\beta) = \psi_1\psi_1^T$ and $\psi_1 = \frac{1}{\sqrt{|V|}}e$, where e^T is the all ones row vector. Therefore, the latter limit is given the van der Waerden matrix $B_* = \frac{ee^T}{|V|}$.

An alternative way to formulate entropy is to quantify the amount of heat flowing through the graph at a particular instant. Given an undirected graph $G = (V, E)$ with unnormalized Laplacian $L = D - A$, the amount of entropy is bounded by the maximum entropy of the Birkhoff-von Neumann decomposition of the heat kernel $K(\beta) = e^{-\beta L}$. This in ensured by the *phase-transition principle* described in [30]: every graph is endowed with a phase-transition point corresponding to the earlier instant where entropy is maximal. In addition, maximal entropy is achieved when the maximum amount of heat is flowing through the graph.

Instantaneous heat flow as defined in [30] is given by the elements of the matrix product $F(\beta) = A : K(\beta)$, where $X : Y = \sum_{ij} X(i,j)Y(i,j)$ is the Frobenius product. Hence, we have that $F(\infty) = A : K(\infty) = A : B_* = \frac{|E|}{|V|}$ is associated with the asymptotic entropy $\log_2(n)$ of the Birkhoff decomposition for the kernel, and the asymptotic (flow) entropy of an undirected graph is

$$H_F = F(\infty) = \frac{|E|}{|V|}. \tag{15}$$

The computational complexity of the asymptotic entropy is quadratic in the graph size, since it needs to visit all the $|V| \times |V|$ entries of the adjacency matrix for G.

Given a directed graph $G_D(V_D, E_D)$, we assume that it it strong connected and aperiodic and with transition matrix $\mathcal{T}$ given by Eq. 8. Otherwise, $\mathcal{T}$ is patched as in Pagerank [38] so that a left eigenvector ϕ_D. exists. The stationary distribution is given by $P_D(i) = \phi_{D;i}$. Then, following [29] we have that the unnormalized directed Laplacian L_D is defined as

$$L_D = \Phi_D^{1/2}\tilde{L}_D\Phi_D^{1/2} = \Phi_D - \frac{\Phi_D\mathcal{T} + \mathcal{T}^T\Phi_D}{2} = \Phi_D - W \ . \tag{16}$$

Since $W(i,j) = (\phi_{D;i}\mathcal{T}(i,j) + \phi_{D;j}\mathcal{T}(j,i))/2$, the role of the weight matrix W is to symmetrize L_D by setting $W(j,i) = \frac{1}{2}\phi_{D;i}\mathcal{T}(i,j) = W(i,j)$ when $d_{out}(j) = 0, d_{out}(i) > 0$, $W(i,j) = \frac{1}{2}\phi_{D;j}\mathcal{T}(j,i) = W(j,i)$ when $d_{out}(i) = 0, d_{out}(j) > 0$, and $W(i,j) = (\phi_{D;i}\mathcal{T}(i,j) + \phi_{D;j}\mathcal{T}(j,i))/2 = W(j,i)$ when $d_{out}(i) > 0, d_{out}(j) > 0$. Therefore, W can be seen as the

weighted adjacency matrix of an undirected graph $G_W = (V_W, E_W)$, where $V_W = V_D$, $E_W = E_D \bigcup \{(i,j) : (j,i) \in E_D\}$ and the weights $W(i,j)$ are associated to the edges. Thus, if the original directed graph G_D is strongly connected, then so is G_W since the latter symmetrization enables alternative paths between the vertices of V_D.

As a result, information diffusion constraints existing in G_D are relaxed in G_W. The constraints are coupled to graph entropy through the phase-transition principle (the harder the constraints the smaller the amount of heat flowing through the graph). The analysis of the heat kernel associated with the directed Laplacian $K_D(\beta) = e^{-\beta L_D} = e^{-\beta(\Phi_D - W)}$ entails the stationary distribution P_D, which is encoded in the diagonal of Φ_D. More precisely, we have

$$\begin{aligned} K_D(\beta) &= e^{-\beta(\Phi_D - W)} \\ &= e^{-\beta\Phi_D}\left(I_{|V_D|} + \beta W + \frac{\beta^2}{2!}W^2 + \frac{\beta^3}{3!}W^3 + \ldots\right) \\ &\approx \sum_{k=0}^{|V_D|^2} W^k \frac{e^{-\beta\Phi_D}\beta^k}{k!}\,, \end{aligned} \tag{17}$$

and W^k is defined in terms of walks of length k:

$$W^k(i,j) = \sum_{\mathcal{S}_k}\prod_{r=1}^{k}\left(\frac{\phi_{D;i_r}\mathcal{T}(i_r,i_{r+1}) + \mathcal{T}(i_{r+1},i_r)\phi_{D;i_{r+1}}}{2}\right)\,, \tag{18}$$

where $\mathcal{S}_k = \{i_1\ i_2\ \ldots i_{k+1}\}$ is a sequence of vertices defining a walk of length k. Therefore $W^k(i,j)$ is the sum of all walks of length k connecting i and j (see [39]). As a result of symmetrization, many of these walks acquire now non-zero probability. For instance, if $(i,j) \in E_D$ but $(j,i) \notin E_D$, there will be a directed path connecting vertices j and i in G_D, since it is strongly connected. However, in G_W we will have $W(j,i) > 0$. Consequently, G_W contributes with many short links. The byproduct of a diffusion process is to create new links, called *transitivity links*. These links (j,i) do not exist in the original graph but are encoded in the components of the heat kernel $K_D^\beta(j,i) > 0$ as β increases. The spectral decomposition of L_D ensures that $\lim_{\beta\to\infty} K_D(\beta) = B_* = \frac{ee^T}{|V_D|}$ as in the undirected case. However, since $\lim_{\beta\to\infty} e^{-\beta\Phi}\beta^k = 0$ for all k and the smaller $\phi(i)$ the less reachable is the i-th vertex in G_D, the components $K_D^\beta(i,j)$ of

the kernel will tend to $\frac{1}{n}$ faster when they correspond to original directed edges. This means that the asymptotic directed flow $F_D(\infty) = A_D : K_D(\infty) = A_D : B_* = \frac{|E_D|}{|V_D|}$ is a good approximation of the entropy trace even for moderate values of β. The reason for this is that it relies on the density of the original directed graph G_D whose edges (and particularly their associated stationary distribution) drive the diffusion process.

Hence, we have that the asymptotic (flow) entropy of a directed graph is

$$H_{FD} = F_D(\infty) = \frac{|E_D|}{|V_D|}. \tag{19}$$

The computational complexity of the asymptotic entropy is quadratic in the number of vertices in the graph G_D.

4. Centroid Expansion Subgraphs

In this section we define a set of centroid expansion subgraphs of an undirected graph. This set will be used for establishing hypergraph complexity traces in Section 5. To commence, we identify a centroid vertex and use this as the root vertex. To this end, for an undirected graph $G(V, E)$, we use Dijkstra's algorithm to compute the shortest path matrix S_G whose element $S_G(i, j)$ represents the shortest path length between vertices v_i and v_j of $G(V, E)$. The average-shortest-path vector S_V for $G(V, E)$ is a vector with the same vertex order as S_G, and with each element $S_V(i) = \sum_{j=1}^{|V|} S_G(i, j)/|V|$ representing the average shortest path from vertex v_i to the remaining vertices. We identify the centroid vertex v_i for $G(V, E)$ as follows

$$\hat{v}_i = \arg\min_i \sum_{j=1}^{|V|} [S_G(i, j) - S_V(i)]^2. \tag{20}$$

The centroid vertex $\hat{v}_i$ of $G(V, E)$ is located by selecting the vertex with the minimum variance of shortest path lengths from all vertices in $G(V, E)$. Therefore, the shortest paths starting from the centroid vertex $\hat{v}_i$ form a *steady* path set that exhibits the least path length variance compared with those path sets originating from the remaining vertices. The vertices surrounding the centroid vertex in $G(V, E)$ lie along different shortest paths from the centroid vertex, and the centroid vertex has a global view of the vertex path length distribution surrounding it. Let $N^K_{\hat{v}_C}$ be a subset

of V satisfying

$$N_{\hat{v}_C}^K = \{u \in V \mid S_G(\hat{v}_C, u) \leq K\}. \tag{21}$$

For a graph $G(V, E)$ with the centroid vertex $\hat{v}_C$, the K-layer centroid expansion subgraph $\mathcal{G}_K(\mathcal{V}_K; \mathcal{E}_K)$ has the vertex set $\mathcal{V}_K$ and edge set $\mathcal{E}_K$ as follows

$$\begin{cases} \mathcal{V}_K = \{u \in N_{v_C}^K\}; \\ \mathcal{E}_K = \{(u, v) \subset N_{v_C}^K \mid (u, v) \in E\}. \end{cases} \tag{22}$$

The number of centroid expansion subgraphs is equal to the greatest length of the shortest path from the centroid vertex to the remaining vertices of the graph.

5. Depth-based Complexity Traces of Hypergraphs

A hypergraph is usually denoted by a pair of sets $HG(V_H, E_H)$ where V_H is a set of vertices and E_H is a set of non-empty subsets of V_H called hyperedges. To obtain hypergraph complexity traces, we first establish a directed line graph using the Perron-Frobenius operator [40, 14]. The reasons for using this graph representation for a hypergraph are twofold. First, pairwise-order representations for hypergraphs allow the graph based complexity analysis to be applied to hypergraphs. Second, the directed line graph avoids the order ambiguities that arise from the straightforward expansion- or clique-based graph representations of a hypergraph [14]. Thus we develop the complexity traces by computing the entropies of a family of centroid expansion subgraphs obtained from the directed line graph.

5.1. Directed Line Graph

The directed line graph of a hypergraph is a dual representation in which each hyperedge is represented by a new vertex. For a hypergraph $HG(V_H, E_H)$, the directed line graph $G_D(V_D, \overrightarrow{E}_D)$ can be established using **Algorithm 1**. Note that, for step 1 there are potential multiple edges between two vertices in $GH(V_G, E_G)$ if the two vertices are encompassed by more than one common hyperedge in $HG(V_H, E_H)$. Suppose there are p hyperedges encompassing two vertices in $HG(V_H, E_H)$. The p hyperedges induce p separated edges between the two vertices in $GH(V_G, E_G)$. For step 3, it is important to stress that unlike the edge set E of an undirect graph

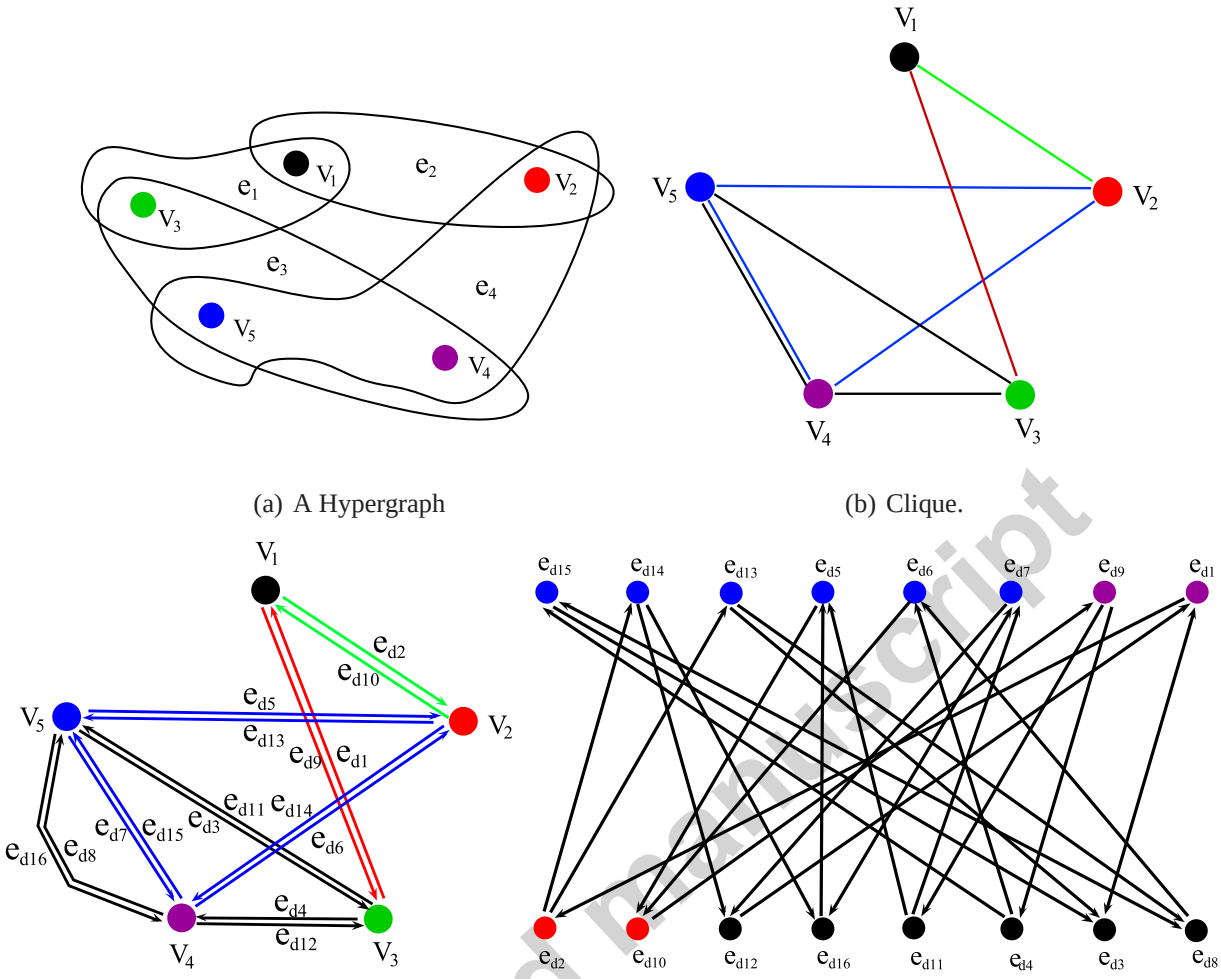


Figure 1: An example of transformation a hypergraph into a directed line graph.

$G(V, E)$, $\overrightarrow{E}_D$ is a set of directed edges of the directed graph $G_D(V_D, \overrightarrow{E}_D)$. The adjacency matrix T_H of $G_D(V_D, \overrightarrow{E}_D)$ is the Perron-Frobenius operator of the original hypergraph. For the (i, j)th entry of T_H, $T_H(i, j)$ is 1 if there is a simple edge directed from the vertex i to the vertex j in the directed line graph, and otherwise it is 0. Unlike the adjacency matrix of an undirected graph, the Perron-Frobenius operator for a hypergraph is not a symmetric matrix. This is because the constraint in Eq.(24) arises in the construction of directed edges. Specifically, any two directed edges induced by the same hyperedge in the original hypergraph are not allowed to establish a directed edge in the directed line graph.

An example of transforming a hypergraph into a directed line graph has been shown in Fig.1.

Algorithm 1: Establishing a directed line graph for a hypergraph

Input: A hypergraph $HG(V_H, E_H)$ where V_H is a set of vertices, and E is a set of non-empty subsets of V_H.

Output: A Perron-Frobenius operator of $HG(V_H, E_H)$ (i.e. the adjacency matrix T_H of a directed line graph $G_D(V_D, \overrightarrow{E}_D)$ for $HG(V_H, E_H)$).

1: Establish the clique expansion graph for $HG(V_H, E_H)$.

- Establish the clique expansion graph $GH(V_G, E_G)$ for $G(V, E)$ by connecting each pair of vertices in e_i through an edge for each hyperedge $e_i \in E$, the vertex and edge sets are

$$\begin{cases} V_G = V; \\ E_G = \{(u, v) \subset e_i \mid e_i \in E\}. \end{cases} \tag{23}$$

2: Establish the associated symmetric digraph for $GH(V_G, E_G)$.

- For $GH(V_G, E_G)$, establish the associated symmetric digraph $DGH(V_G, E_d)$ by replacing each edge of $GH(V_G, E_G)$ by a directed edge pair in which the two directed edge are inverse to each other.

3: Establish the directed line graph of $GH(V_G, E_G)$ through $DGH(V_G, E_d)$.

- Establish the directed line graph $G_D(V_D, \overrightarrow{E}_D)$ of $HG(V_H, E_H)$ based on $DGH(V_G, E_d)$. The vertex set V_D and edge set $\overrightarrow{E}_D$ of the $G_D(V_D, \overrightarrow{E}_D)$ are defined as

$$\begin{cases} V_D = E_d; \\ \overrightarrow{E}_D = \{(u, v)_i, (v, w)_j \in E_d \times E_d \mid i \neq j\}. \end{cases} \tag{24}$$

where the subscripts i and j denote the indices of the hyperedges from which the directed edges (u, v) and (v, w) are induced respectively.

For the example hypergraph $HG(V_H, E_H)$ shown in Fig.1(a), the clique graph $GH(V_G, E_G)$ is shown in Fig.1(b). In $GH(V_G, E_G)$, the edges belonging to the common clique are indicated by the same colour while the different cliques are coloured differently. Furthermore, there are two different edges between v_4 and v_5, and these edges are induced by the hyperedge e_3 and e_4 of $HG(V_H, E_H)$, respectively. The associated symmetric digraph $DGH(V_G, E_d)$ of $GH(V_G, E_G)$ is shown in Fig.1(c), and the resulting directed line graph $G_D(V_D, \overrightarrow{E}_D)$ from $DGH(V_G, E_d)$ is shown in Fig.1(d).

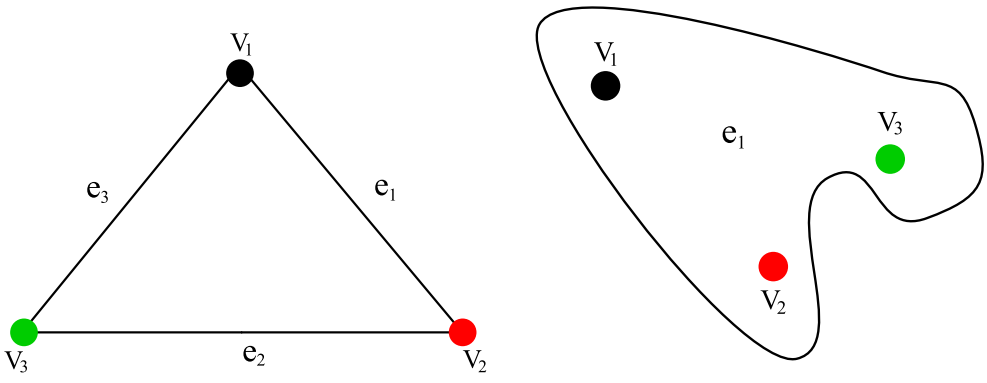
The transformation of the hypergraph $HG(V_H, E_H)$ into the directed line graph $G_D(V_D, \overrightarrow{E}_D)$ requires time complexity $O(|V_D|^2)$. This is because the construction of the adjacency matrix of $G_D(V_D, \overrightarrow{E}_D)$ relies on visiting all the $|V_D|$ ($|V_D| = |E_d|$) edges in $DGH(V_G, E_d)$ and establishing all $|V_D|^2$ entries in the incidence matrix of G_D.

5.2. Theoretical Properties

The directed line graph and its Perron-Frobenius operator have several interesting properties.

a) Compared to the (hyper)graph adjacency or Laplacian matrix, the Perron-Frobenius operator spans a higher dimensional feature space where it may expose richer (hyper)graph characteristics. This property is a result of the fact that the cardinality of the vertex set for the directed line graph is much greater than, or at least equal to, that of the original (hyper)graph. Hence, the adjacency matrix (i.e. the Perron-Frobenius operator) of the directed line graph is described in a higher dimensional space than the original (hyper)graph.

b) The directed line graph represents a (hyper)graph in a complete manner such that it naturally avoids the information loss arising in the spectral truncation [10] or the clique graph approximation [2]. This property is due to the constraint in Eq.(24), i.e., the connecting edge pair induced by the same hyperedge in the original hypergraph cannot establish a directed edge in the directed line graph. Actually this induces a bi-partition in the vertices of V_D. In other words, such a directed line graph can distinguish different edges derived from the same hyperedge. This property is illustrated in Fig.1(d). On the other hand, the clique expansion graph $GH(V_G, E_G)$ from the original hypergraph $HG(V_H, E_H)$ only records adjacency relationships between vertex pairs of the hypergraph, and cannot distinguish whether or not two edges are derived from the same hyperedge.



(a) (b)

Figure 2: Hypergraph examples.

This property is illustrated in Fig.1(b). Hence, for two different hypergraphs (e.g., the hypergraphs shown in Figs.2(a) and (b)) they may have the same clique expansion graph, and thus the same resulted adjacency and Laplacian matrices resulting from the clique expansion graph. On the other hand, the directed line graph defined in Eq.(24) may still produce total different structures for the two hypergraphs. In Fig.2(b) we have an unique hyperedge e_1 that encodes the same clique which defines the graph in Fig.2(a).

These properties indicate that the direct line graph and its Perron-Frobenius operator can offer us an elegant way for hypergraph complexity analysis which can not only capture precise hypergraph complexity information but can also reflect richer characteristics of hypergraphs.

5.3. Depth-based Complexity Traces of Hypergraphs

We define a depth-based complexity trace for a hypergraph based on its directed line graph. Simply establishing subgraphs with increasing layer size along the shortest paths on a directed line graph tends to ignore certain topological information. The reason for this is that a path may not exist between two given vertices in a connected directed line graph. To overcome this problem, we identify the centroid vertex for the undirected line graph of a hypergraph. The undirected line graph $G_U(V_U, E_U)$ of hypergraph $HG(V_H, E_H)$ can be obtained by replacing each pair of inversely directed edges in $G_{DL}(V_{DL}, \overrightarrow{E}_{DL})$ by an undirected edge. Then we develop two classes of complexity traces for $HG(V_H, E_H)$, which we refer to as the undirected complexity trace and directed complexity trace respectively.

Definition 1 (Undirected complexity trace) For a hypergraph $HG(V_H, E_H)$ and its undirected line graph $G_U(V_U, E_U)$, the undirected complexity trace CT^U is an $L^{\max}$ dimensional vector

$$CT^U = [H(\mathcal{G}_{\mathcal{U}1}), \cdots, H(\mathcal{G}_{\mathcal{U}K}), \cdots, H(\mathcal{G}_{\mathcal{U}L^{\max}})]^T. \tag{25}$$

where $L^{\max}$ is the greatest length of the shortest paths from the centroid vertex $\hat{v}_C^U$ to the remaining vertices in $G_U(V_U, E_U)$, $\mathcal{G}_{\mathcal{U}K}$ is the K-layer centroid expansion subgraph of $G_U(V_U, E_U)$, and $H(\mathcal{G}_{\mathcal{U}K})$ is the entropy of $\mathcal{G}_{\mathcal{U}K}$. □

Here the entropy function $H(\cdot)$ could be either the von Neumann entropy $H_{VN}(\cdot)$ given in Eq.(4) or the Shannon entropy $H_S(\cdot)$ given in Eq.(6).

Next we describe how to extend these ideas to a directed complexity trace for the hypergraph $HG(V_H, E_H)$. For the directed line graph $G_{DL}(V_{DL}, \overrightarrow{E}_{DL})$ of $HG(V_H, E_H)$, it is impossible to establish a K-layer centroid expansion subgraph according to Eq.(22), because the edges of $G_{DL}(V_{DL}, \overrightarrow{E}_{DL})$ are directed. For a hypergraph $HG(V_H, E_H)$, given the K-layer centroid expansion subgraph $\mathcal{G}_{\mathcal{U}K}(\mathcal{V}_{\mathcal{U}K}; \mathcal{E}_{\mathcal{U}K})$ of $G_U(V_U, E_U)$, we develop a K-layer pseudo centroid expansion subgraph $\mathcal{G}_{\mathcal{D}K}(\mathcal{V}_{\mathcal{D}K}; \overrightarrow{\mathcal{E}}_{\mathcal{D}K})$ of $G_{DL}(V_{DL}, \overrightarrow{E}_{DL})$ with vertex and edge sets as follows

$$\begin{cases} \mathcal{V}_{\mathcal{D}K} = \mathcal{V}_{\mathcal{U}K}; \\ \overrightarrow{\mathcal{E}}_{\mathcal{D}K} = \{(u, v) \in \overrightarrow{E}_D \mid (u, v) \in \mathcal{E}_{\mathcal{U}K}\}. \end{cases} \tag{26}$$

Note that there is a strict order for any pair of vertices $(u, v) \in \overrightarrow{\mathcal{E}}_{\mathcal{D}K}$.

Definition 2 (Directed complexity trace) For a hypergraph $HG(V_H, E_H)$ together with its directed line graph $G_{DL}(V_{DL}, \overrightarrow{E}_{DL})$ and undirected line graph $G_U(V_U, E_U)$, the directed complexity trace CT^D is an $L^{\max}$ dimensional vector defined as

$$CT^D = [H_D(\mathcal{G}_{\mathcal{D}1}), \cdots, H_D(\mathcal{G}_{\mathcal{D}K}), \cdots, H_D(\mathcal{G}_{\mathcal{D}L^{\max}})]^T. \tag{27}$$

where $L^{\max}$ is the greatest length of the shortest paths from the centroid vertex $\hat{v}_C^U$ to the remaining vertices in $G_U(V_U, E_U)$, $\mathcal{G}_{\mathcal{D}K}$ is the K-layer pseudo centroid expansion subgraph of $G_{DL}(V_{DL}, \overrightarrow{E}_{DL})$, and $H_D(\mathcal{G}_{\mathcal{D}K})$ is the entropy of the directed subgraph $\mathcal{G}_{\mathcal{D}K}$. □

Based on the definition in Section 5.1, the directed line graph of a (hyper)graph is a strongly directed graph. Hence the entropy function $H_D(\cdot)$ should be the strongly directed von Neumann entropy $H_{VN}^D(\cdot)$ in Eq.(14).

Hypergraphs of Different Sizes: Note that, the L^{max} layer expansion subgraph is the undirected line graph itself. The dimension of a hypergraph complexity trace vector is thus equal to the greatest layer L^{max}. However, the complexity trace vectors for hypergraphs of different sizes may exhibit various lengths. To compare these hypergraphs by using complexity trace vectors, we need to make the vector lengths uniform. This is achieved by padding out the dimensions of the complexity trace vectors. Hence, for complexity trace vectors CT_i and CT_j of the two hypergraphs HG_i and HG_j with dimensions L_p and L_q respectively, where $L_p > L_q$, we use the L_q-th element value of CT_j as the added padding value for the extended $L_q + 1$-th to L_p-th elements of CT_j.

5.4. Disscussions of the Hypergraph Complexity Traces

The two proposed depth-based complexity traces possess the following key features. **a)** Eq.(4) indicates that the von Neumann entropy H_{VN} is associated with the degrees of connected vertices. Accordingly, the undirected complexity trace CT^U is sensitive to changes of edge structures (e.g. edge deletions) associated with vertices of low degrees in $G_U(V_U, E_U)$. Such edges usually form bridges between vertex clusters in $G_U(V_U, E_U)$. Hence, the proposed undirected complexity trace CT^U associated with the von Neumann entropy H_{VN} is sensitive to the interconnections between vertex clusters within $G_U(V_U, E_U)$. **b)** Eq.(6) indicates that for the Shannon entropy H_S vertices with large degrees dominate the value of the entropy. Hence, the undirected complexity trace CT^U associated with H_S is suited to characterizing hypergraphs with strongly intra-connected structures. **c)** Eq.(14) indicates that the von Neumann entropy H_{VN}^{SD} depends on the in-degree and out-degree statistics of each pair of vertices linked by an edge. Hence, the directed complexity trace CT^D associated with H_{VN}^{SD} is sensitive to the in-degree and out-degree of each pair of vertices connected within $G_{DL}(V_{DL}, \overrightarrow{E}_{DL})$. **d)** As a result of the properties of a directed line graph stated in Section 5.1, the proposed complexity traces from line graphs can reflect precisely the rich complexity information for both uniform and nonuniform hypergraphs. **e)** Eq.(25) and Eq.(27) indicate that the depth-based complexity traces provide a multi-dimensional complexity characterization via the increasing layer substructures of the line graph from the centroid vertex.

Furthermore, since a hypergraph is a generalization of a graph and a graph can also be transformed into a directed line graph, the construction of the complexity trace for a graph is just a

special case of our hypergraph method. On the other hand, the complexity trace for a graph can be directly constructed from the original graph by identifying its centroid vertex and establishing the centroid expansion subgraphs on it (e.g., the depth-based complexity traces for graph defined in [32]). However, the proposed complexity traces for a graph through its line graph can capture richer characteristics of complexity than those obtained from the original graph, because the Perron-Frobenius operator can extract (hyper)graph characteristics in a higher dimensional feature space than that of the original (hyper)graph. The proposed complexity traces for (hyper)graphs focus on measuring how the entropy based complexities of their subgraphs from the line graphs (i.e. graphs transformed from the original (hyper)graphs) vary with increasing layer size. Such complexity traces reflect high dimensional depth-based complexity characteristics of (hyper)graphs and can be used for (hyper)graph clustering or classification. By contrast, the depth-based complexity measure in [30], the Shannon entropy measures in [21] and the von-Neumann entropy measure in [26] are based on the global structure of the original graph, and only provide an unidimensional complexity characterization.

5.5. Analysis of Computational Complexity

Suppose the line graph, either $G_{DL}(V_{DL}, \overrightarrow{E}_{DL})$ or $G_U(V_U, E_U)$, extracted from $HG(V_H, E_H)$ has n vertices. The computational complexities for constructing the proposed complexity traces for $HG(V_H, E_H)$ are governed by the following processes. 1) The construction of the centroid expansion subgraphs which involves using Dijkstra algorithm to compute the shortest path matrix to locate the centroid vertex and implementing the transformation from the hypergraph $HG(V_H, E_H)$ into the line graph. The Dijkstra algorithm takes time $O(n^2)$. The transformation to the line graph has time complexity $O(n^2)$. As a result the construction of the representation has time complexity $O(n^2)$. 2) The computations of a) the von Neumann entropy in Eq.(4) and b) the Shannon entropy in Eq.(6) for the centroid expansion subgraphs from $G_U(V_U, E_U)$ (i.e., for the undirected complexity trace), or c) the von Neumann entropy in Eq.(14) and d) the asymptotic (flow) entropy in Eq.(19) for the centroid expansion subgraphs from $G_{DL}(V_{DL}, \overrightarrow{E}_{DL})$ (i.e., for the directed complexity trace). Through the definitions in Sections 2, 3 and 4, these entropies all require time complexity $O(n^2 L^{max})$.

L^{max} is equal to the greatest length of shortest paths from the centroid vertex of $G_U(V_U, E_U)$, and $L^{max} < n$. Therefore, the worst-case time complexities of our undirected and directed complexity traces for $HG(V_H, E_H)$ using the four required entropies are all $O(n^3)$.

As a result, our depth-based complexity traces can be computed in polynomial time. The reason for this is that we efficiently compute the required entropies on a small set of expansion subgraphs rooted at the centroid vertex of a line graph. By contrast, the depth-based complexity measure described in [30] establishes expansion subgraphs for each vertex of a given undirected graph (e.g. a graph having n vertices) and then computes the intrinsic complexities on the subgraphs. It hence requires time complexity $O(n^7)$.

6. Experimental Evaluations

6.1. Hypergraph and Graph Datasets

We demonstrate the performance of our complexity traces on several (hyper)graph datasets. We use a hypergraph based dataset abstracted from the COIL image database and five standard graph based datasets abstracted from bioinformatics databases [30, 43, 44, 42] for the experimental evaluations. These datasets are COIL (for hypergraphs), MUTAG, CATH1, CATH2 and PPIs (for graphs).

COIL: The COIL database consists of images of 100 3D objects. In our experiments, we use selected images for three similar cups, three similar bottles and three pieces of similar vegetables. For each object we employ 18 images captured from different viewpoints. The hypergraphs are abstracted using the feature hypergraph method [14]. Details about the feature hypergraph method can be found in [14]. The maximum, minimum and average vertices of the COIL dataset are 549, 213 and 412.5 respectively.

MUTAG: The MUTAG dataset consists of graphs representing 188 chemical compounds. The maximum, minimum and average number of vertices are 28, 10 and 17.93 respectively. The edges of each compound are labeled with a real number, we transform these graphs into unweighted graphs.

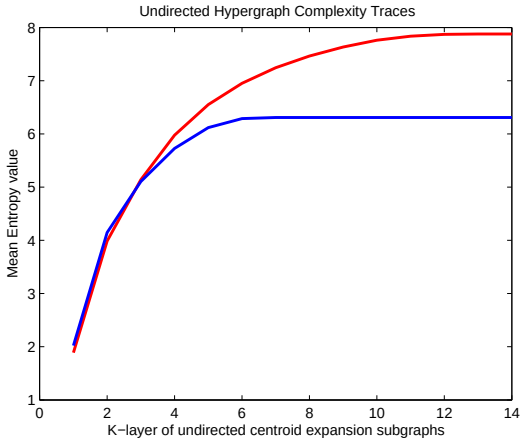
CATH1 and CATH2: The CATH1 dataset consists of proteins in the same class (i.e Mixed Alpha-Beta), but has different architectures (i.e. Alpha-Beta Barrel vs. 2-layer Sandwich). CATH2 has

proteins in the same class (i.e. Mixed Alpha-Beta), architecture (i.e. Alpha-Beta Barrel), and topology (i.e. TIM Barrel), but in different homology classes (i.e. Aldolase vs. Glycosidases). The CATH2 dataset is harder to classify, since proteins in the same topology class are structurally similar. The protein graphs are 10 times larger in size than chemical compounds, with $200 - 300$ vertices. There are 712 and 190 testing graphs in the CATH1 and CATH2 datasets.

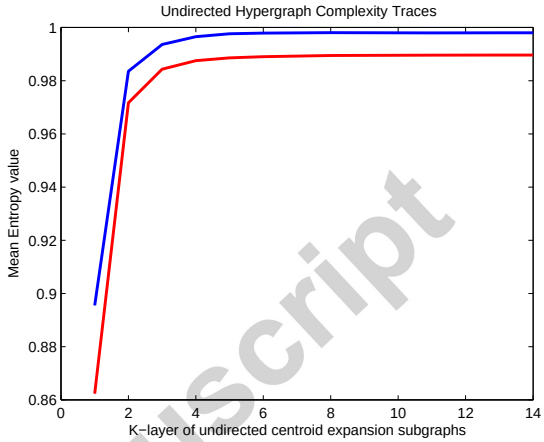
PPIs: The PPIs dataset consists of protein-protein interaction networks (PPIs). The graphs describe the interaction relationships between histidine kinase in different species of bacteria. Histidine kinase is a key protein in the development of signal transduction. If two proteins have direct (physical) or indirect (functional) association, they are connected by an edge. There are 219 PPIs in this dataset and they are collected from 5 different kinds of bacteria. We select *Proteobacteria*40 PPIs and *Acidobacteria*46 PPIs as the testing graphs. The maximum, minimum and average number of vertices of the selected testing graphs are 232, 3 and 109.60 respectively.

6.2. Evaluation of Interior Complexity Traces

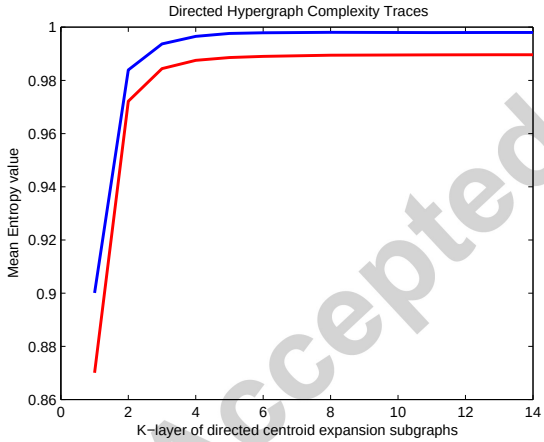
We commence by illustrating the representational power of the proposed complexity traces for hypergraphs. We demonstrate that they can be used to distinguish different hypergraphs. The evaluation utilizes 36 hypergraphs abstracted separately from the images of two different objects, namely a box and a cup in the COIL image database. For each object we use 18 images captured from different viewpoints. The hypergraphs for individual images are established by using the feature hypergraph method. For each hypergraph, we locate the centroid vertex of its (un)directed line graph, and construct the proposed complexity traces. Figs.3(a), (b), (c) and (d) show the mean values of the undirected complexity traces using the Shannon entropy (UCTS) and the von Neumann entropy (UCTV), together with the directed complexity traces using the von Neumann entropy (DCTV) and the asymptotic (flow) entropy (DCTA), respectively. In Fig.3 the x-axis represents the order of the K-layer centroid expansion subgraph for each hypergraph, while the y-axis represents the mean entropy value as a function of the expansion subgraph order. Here the blue and red lines represent the mean entropy values of the complexity traces for the hypergraphs abstracted from the box and cup objects respectively. The main feature to note is that the mean entropy values for the different objects are quite dissimilar.



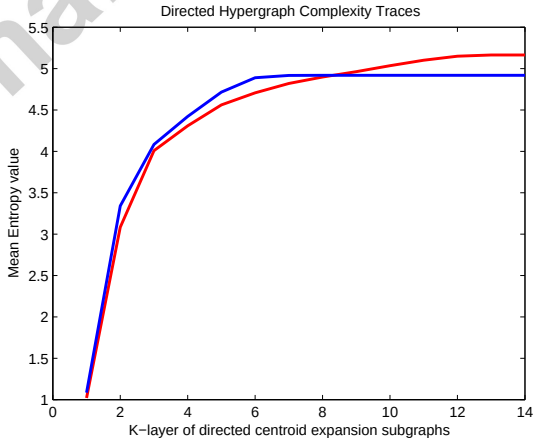
(a) For UCTS



(b) For UCTV



(c) For DCTV



(d) For DCTA

Figure 3: Mean Entropy Values of Complexity Traces.

6.3. Experiments on Image Hypergraphs

6.3.1. Experimental setup

We illustrate the performance of our proposed complexity traces UCTS, UCTV, DCTV and DCTA on a hypergraph classification problem. The hypergraph dataset for testing is again abstracted from the COIL image database. We also compare our methods with several alternative state-of-the-art hypergraph based learning methods. These methods include 1) the Ihara coefficients for hypergraphs (HCIZF) [14], 2) the truncated Laplacian spectra (TLS) and truncated normalized Laplacian spectra (TNLS) [2]. We compute the feature vectors of test hypergraphs using both our own methods and the alternatives. We then perform 10-fold cross-validation using the Support Vector Machine (SVM) Classifier associated with the Sequential Minimal Optimization (SMO) [45] and the Pearson VII universal kernel (PUK) [46] to compute the classification accuracies. We use nine samples for training and one for testing. All the SMO-SVMs and their parameters were performed and optimized on a Weka workbench [46]. To exclude random effects of fold assignments, we repeat the the whole experiments 10 times. We report the average classification accuracy in Table I.

6.3.2. Experimental Results and Evaluations

From Table 1 it is clear that our methods achieve the greatest accuracies over all image datasets. **1)** Our UCTS, UCTV, DCTV and DCTA methods outperform TLS and TNLS which both use spectral information for the hypergraphs. The reason for this is that our methods based on the line graph of a (hyper)graph can capture richer (hyper)graph characteristics than the (hyper)graph spectral representations. They also avoid the spectral truncation arising in TLS and TNLS. **2)** For the hypergraphs extracted from the images of the cup object, the maximum and minimum number of vertices are 310 and 213 respectively. Here the accuracy of HCIZF is competitive with that of our complexity traces. Like our complexity traces, HCIZF also relies on directed line graphs, and exploits richer (hyper)graph characteristics. However, for the hypergraphs extracted from the images of the bottle and vegetable objects, where the maximum and minimum number of vertices are 549 and 305 respectively, HCIZF is intractable for characterizing the hypergraph structures. The reason for this is that the computation of the underlying Ihara coefficients tends to result in

Table 1: Experimental Comparisons on Hypergraphs

Datasets	Cups	Bottles	Vegetables
UCTS	**100**	**100**	**100**
UCTV	**100**	**100**	**100**
DCTV	**100**	**100**	**100**
DCTA	**100**	**100**	**100**
TLS	92.31	83.44	82.91
TNLS	55.27	90.00	71.96
HCIZF	**100**	–	–

infinities even for hypergraphs of moderate sizes. In contrast, our proposed complexity traces can easily scale to large hypergraphs, and our experimental results verify this advantage.

6.4. Experiments on Graphs

6.4.1. Experimental setup

We evaluate the performance of our proposed complexity traces UCTS, UCTV and DCTV on a graph classification problem. The datasets for testing are abstracted from bioinformatics databases. We also compare our methods with alternative state-of-the-art graph based learning methods. The comparative methods include 1) the Weisfeiler-Lehman subtree kernel (WL) [47], 2) the von-Neumann thermodynamic depth complexity (VNTD) [30], 3) the von-Neumann graph entropy (VNGE) [26], 4) the Shannon graph entropy (SGE) defined in Eq.(6), 5) the Shannon entropies associated with the information functionals f^V (FV) and f^P (FP) [21], 6) the Ihara coefficients for graphs (GCIZF) [14], and 7) the depth-based complexity traces of graphs [32] computed using the Shannon entropy associated with both the steady state random walk (ECTS) and the von Neumann entropy (ECTV). For the Weisfeiler-Lehman subtree kernel we compute the kernel matrix of each dataset, and then perform Principle Component Analysis (PCA) on the kernel matrix to embed graphs into a feature space. For the remaining methods, we calculate the feature vectors or feature values of testing graphs. We then perform 10-fold cross-validation using the SMO-SVMs described in Section 6.3 to compute the classification accuracies for each of the methods in turn. We report the average classification accuracy over the 10-fold cross validation for each method in Table 2. We also report the runtime of each method in Table 2. The runtime is

evaluated under Matlab R2011a running on an Intel(i5) 2.5GHz 2-Core processor (i.e. i5-3210M).

6.4.2. Experimental Results and Evaluations

From Table 2, we can obtain the following conclusions. **1)** On the MUTAG, CATH1 and PPIs datasets, our complexity trace UCTS outperforms all the alternative methods. The complexity traces UCTV, DCTV and DCTA outperform or are competitive to the alternative methods. On the CATH2 dataset, our complexity trace UCTS outperforms all the alternative methods, excluding the ECTV. The complexity traces UCTV, DCTV and DCTA outperform or are competitive to the alternative methods. Key to the effectiveness of our methods is that our hypergraph complexity traces probe a graph using the line graph, and can thus reflect richer graph characteristics in a higher dimensional feature space. On the other hand, the alternative methods are based on the original graph representation. In particular, the entropy based complexity measures (i.e. VNGE, SGE, FV and FP) are simply computed based on the global structure of the original graph, and only provide an uni-dimensional complexity characterisation. **2)** Although GCIZF is also based on a line graph representation, it is outperformed by our complexity trace methods on each of the datasets studied. This is because the centroid expansion subgraphs allow our methods to capture a depth-based information that GCIZF cannot convey. **3)** The runtime of our complexity trace methods is clearly faster than that of the alternative depth-based complexity method VNTD. It is also competitive with GCIZF, the fast subtree kernel WL and the fast entropy measures VNGE, SGE, FV and FP. The reason for this efficiency is that the required graph entropies in our methods can all be computed in polynomial time. Compared to the depth-based graph complexity measure VNTD, our complexity trace methods avoid either establishing the expansion subgraphs from each vertex or computing the intrinsic complexities on the subgraphs. **4)** Generally speaking, the accuracies of our UCTS, UCTV and DCTV methods are very similar on the MUTAG, CATH1 and CATH2 datasets. However, the accuracies of our UCTS method are obviously higher than those of our UCTV and DCTV on the PPIs dadaset. The reason for this is that the entropy value of an (un)directed graph computed using either the undirected or the directed von Neumann entropy tends to be close to 1. This implies that the Shannon entropy is better suited for distinguishing graphs of different classes than the von Neumann entropy. **5)** The accuracies of the proposed

complexity traces with the entropies VNGE and SGE are obviously greater than those based on the original entropies. This verifies again that our complexity trace methods capture richer structural characteristics than the original graph based methods. **6)** The accuracy of the DCTV method is generally greater than that of the UCTV method, because DCTV considers directional information residing on the edges of a line graph. However, UCTV ignores these edge directions. This also implies that the performance of our complexity traces also depends on that of the required graph entropy. Generally speaking, the hypergraph complexity traces computed using the Shannon or von Neumann entropy (i.e., the UCTS or UCTV) outperform the graph complexity traces using the same entropy (i.e., ECTS or ECTV). The reason for this is that the directed line graph obtained by transforming the original graph can capture rich structural characteristics. This indicates that the complexity traces from the line graph reflect deeper complexity information than those obtained from the original graph. **7)** Finally, for our DCTA method the accuracies on the MUTAG and PPIs datasets are very good, but a little lower on the CATH1 and CATH2 datasets. The main reason for this is that the edge density of the line graph is related to that of the original graph. The graphs in the MUTAG and PPIs datasets are very sparse, with average edge density close to 2.19. However, for the CATH1 and CATH2 datasets, the average edge density is much larger, and is about 8.2. Since the required asymptotic (flow) complexity for the DCTA method is $|E_D|/|V_D|$, the complexities for these (sub)graphs are thus similar. This indicates that the complexity measure may not be suitable for graphs having high edge density, but perform well on sparse graphs.

7. Conclusion

In this paper, we have shown how to construct depth-based complexity traces for a hypergraph. Our methods are based on transforming a hypergraph into a directed line graph. This not only accurately reflects the multiple relationships exhibited by the hypergraph, but is also amenable to complexity analysis. By neglecting the directed edges of the directed line graph, we have identified a centroid vertex, and thus obtained a family of expansion subgraphs around the vertex with increasing layer size. The complexity traces of a hypergraph have been characterized by measuring how the required entropies of these subgraphs vary with increasing layer size. Experiments demonstrate the effectiveness and efficiency of our methods.

Table 2: Experimental Comparisons on Graphs

Datasets	MUTAG	PPIs	CATH1	CATH2
UCTS	**87.23**	**83.72**	**98.87**	78.94
UCTV	86.17	75.58	98.45	77.89
DCTV	86.17	76.21	98.79	78.94
DCTA	86.17	79.09	92.13	72.63
ECTS	86.35	74.41	**98.87**	78.42
ECTV	84.57	70.93	98.73	**80.47**
VNTD	83.51	67.44	–	–
VNGE	85.10	63.95	98.45	75.78
SGE	85.10	67.44	98.17	76.31
FV	84.57	70.93	96.76	76.31
FP	85.63	70.93	96.91	76.31
WL	84.57	73.25	98.17	73.15
GCIZF	80.85	70.93	–	–

Datasets	MUTAG	PPIs	CATH1	CATH2
UCTS	1”	45”	9′15”	6′12”
UCTV	1”	45”	9′15”	6′12”
DCTV	1”	53”	16′32”	10′50”
DCTA	1”	39”	8′55”	5′39”
ECTS	1”	1”	5”	2”
ECTV	1”	1”	5”	2”
VNTD	19′53”	49′50”	$> 1day$	$> 1day$
VNGE	1”	1”	1”	1”
SGE	1”	1”	1”	1”
FV	1”	1”	12”	5”
FP	1”	1”	12”	5”
WL	1”	1”	2′41”	51”
GCIZF	1”	52”	–	–

Our future plans are to extend the work in a number of ways. First, in prior work we have developed methods for characterising graphs using the commute time [48] and the heat kernel [49]. Both the commute time and the heat kernel of an undirected graph encapsulate the path length information between vertices. It would be interesting to use the commute time or heat kernel as a means of identifying a centroid vertex. Second, in [52] Haussler has proposed a generic method, referred as R-convolution, to define a kernel between two graphs by decomposing them and measuring the pairwise similarities between the resulting substructures. Examples include graph kernels based on all pairs of a) walks [50], b) paths [51] and c) restricted subgraph or subtree structures [47]. It would be interesting to use the expansion subgraphs defined in this paper as a new type of depth-based (hyper)graph decomposition to define a novel (hyper)graph kernel. Finally, in [53] we have explored the use of the discrete-time quantum walks on the directed line graph, which can be constructed by transforming a hypergraph. It would be interesting to extend this work, using the discrete-time quantum walks to compute the von Neumann entropy associated with a quantum state. This may provide a more principled means of computing a quantum depth-based complexity trace of a hypergraph.

Acknowledgments

This work is supported by National Natural Science Foundation of China (Grant No. 61503422). Francisco Escolano is supported by the project TIN2012-32839 of the Spanish Government. Edwin R. Hancock is supported by a Royal Society Wolfson Research Merit Award.

[1] S. Agarwal, J. Lim, L. Zelnik-Manor, P. Perona, D. Kriegman, and S. Belongie, "Beyond pairwise clustering," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 838-845.

[2] D. Zhou, J. Huang, and B. Schölkopf, "Learning with hypergraphs: clustering, classification, and embedding," in *Proceedings of Advanced Neural Information Processing System*, 2007, pp. 1601-1608.

[3] G. Wachman, and R. Khardon, "Learning from interpretations: a rooted kernel for ordered hypergraphs," in *Proceedings of International Conference on Machine Learning*, 2007, pp. 943-950.

[4] R. Zass, and A. Shashua, "Probabilistic graph and hypergraph matching," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp.1-8.

[5] O. Duchenne, F.R. Bach, I.S. Kweon, and J. Ponce, "A tensor-based algorithm for high-order graph matching," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 1980-1987.

[6] A. Shashua, R. Zass, and T. Hazan, "Multi-way clustering using super-symmetric non-negative tensor factorization," in *Proceedings of European Conference on Computer Vision*, 2006, pp. 595-608.

[7] A. Shashua, and A. Levin, "Linear image coding for regression and classification using the tensor-rank principle," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2001, pp. 623-630.

[8] Z. He, A. Cichocki, S. Xie, and K. Choi, "Detecting the number of clusters in n-way probabilistic clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 2006-2012, 2010.

[9] V.M. Govindu, "A tensor decomposition for geometric grouping and segmentation," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 1150-1157.

[10] G. Chen, and G. Lerman, "Spectral curvature clustering," *International Journal of Computer Vision*, vol. 81, pp. 317-330, 2009.

[11] G. Chen, and G. Lerman, "Foundations of a multi-way spectral clustering framework for hybrid linear modeling," *Journal of Foundations of Computational Mathematics*, vol. 9, pp. 517-558, 2009.

[12] G. Chen, S. Atev, and G. Lerman, "Kernel spectral curvature clustering," in *Proceedings of ICCV Workshop on Dynamical Vision*, 2009, 317-330.

[13] S.R. Bulo, and M. Pelillo, "A game-theoretic approach to hypergraph clustering," in *Proceedings of Advanced Neural Information Processing System*, 2009, pp. 1571-1579.

[14] P. Ren, T. Aleksic, R.C. Wilson, and E.R. Hancock, "A polynomial characterization of hypergraphs using the Ihara zeta function," *Pattern Recognition*, vol.44, pp. 1941-1957, 2011.

[15] D.P. Feldman, and J.P. Crutchfield, "Measures of statistical complexity: Why?," *Physics Letters A*, vol. 238, pp. 244252, 1998.

[16] J. Körner, "Coding of an information source having ambiguous alphabet and the entropy of graphs," in *Proceedings of Transactions of of the 6th Prague Conference on Information Theory*, 1973, pp. 411-425.

[17] A. Mowshowitz, "Entropy and the complexity of the graphs I: An index of the relative complexity of a graph," in *Bulletin Mathematical Biophys*, vol. 30, pp. 175-204, 1968.

[18] N. Rashevsky, "Life, information theory, and topology," in *Bulletin Mathematical Biophys*, vol. 17, pp. 229-235, 1955.

[19] E. Trucco, "A note on the information content of graphs," in *Bulletin Mathematical Biophys*, vol. 18, pp. 129-135, 1956.

[20] M. Dehmer, "Information processing in complex network: Graph entropy and information functionals," *Applied Mathematics and Computing*, vol. 201, pp. 82-94, 2008.

[21] M. Dehmer, and A. Mowshowitz, "A history of graph entropy measures," *Information Sciences*, vol. 181, pp. 57-78, 2011.

[22] F. Passerini, and S. Severini, "Quantifying complexity in networks: The von Neumann entropy," *International Journal of Agent Technologies and Systems*, vol.1, pp. 58-67, 2009.

[23] K. Anand, G. Bianconi, and S. Severini, "Shannon and von Neumann entropy of random networks with heterogeneous expected degree," *Physical Review E*, vol. 83, pp. 036109, 2011.

[24] S.L. Braunstein, S. Ghosh, and S. Severini, "The Laplacian of a graph as a density matrix: A basic combinatorial approach to separability of mixed states," *Annals of Combinatorics*, vol. 10, pp. 291-317, 2006.

[25] C.D. Godsil, G. Royle, and C.D. Godsil, "Algebraic graph theory," *Springer*, New York, 2001.

[26] L. Han, F. Escolano, and E.R. Hancock, "Graph characterizations from von Neumann entropy," Pattern Recognition Letters 33(15): 1958-1967 (2012)

[27] B. Bonev, L.Chuang, F. Escolano, "How do image complexity, task demands and looking biases influence human gaze behavior," Pattern Recognition Letters 34(7): 723-730 (2013)

[28] C. Ye, R.C. Wilson, C.H. Comin, L.F. Costa, and E.R. Hancock, "Approximate von Neumann entropy for directed graphs," Physic Review E 89: 052804 (2013).

[29] F. Chung, "Laplacians and the Cheeger Inequailty for Directed Graphs," *Annals of Combinatorics*, vol. 9, pp. 1-19, 2005.

[30] F. Escolano, E.R. Hancock, and M.A. Lozano, "Heat diffusion: Thermodynamic depth complexity of networks," *Physical Review E*, vol. 85, pp. 206236, 2012.

[31] J.P. Crutchfield and, C.R. Shalizi, "Thermodynamic depth of causal states: Objective complexity via minimal representations," *Physical Review E*, vol. 59, pp. 275283, 1999.

[32] L. Bai and, E.R. Hancock, "Depth-based complexity traces of graphs," *Pattern Recognition*, vol. 47(3), pp. 1172-1186, 2014.

[33] L. Bai, E.R. Hancock, L. Han and, P. Ren, "Graph clustering using graph entropy complexity traces," *Proceedings of International Conference on Pattern Recognition*, 2012, pp. 2881-2884.

[34] L. Bai and, E.R. Hancock, "Graph Complexity from the Jensen-Shannon Divergence," in *Proceedings of SSPR/SPR*, 2012, pp.79-88.

[35] L. Bai, and E.R. Hancock, "Graph Kernels from the Jensen-Shannon Divergence," in *Journal of Mathematical Imaging and Vision*, vol. 47(1-2), pp. 60-69, 2013.

[36] F. Passerini, and S. Severini, "The von Neumann entropy of networks," *International Journal of Agent Technologies and Systems*, pp. 58-67, 2008.

[37] R. Kondor and J. Lafferty, "Diffusion Kernels on Graphs and Other Discrete Input Spaces" in *Proceedings of ICML* 2002.

[38] L. Page, S. Brin, R. Motwani, T. Winograd, "The PageRank citation ranking: Bring order to the web." (Technical Report). Stanford University, 1998.

[39] Brasseur, C.E., Grady, R.E., Prassidis, S., "Coverings, Laplacians and heat kernels of directed graphs." Electr. J. Comb 01/2009; 16(1), 2009.

[40] C.K. Storm, "The zeta function of a hypergraph," *Electronic Journal of Combinatorics*, vol. 13, pp. 1-26, 2006.

[41] H. Bass, "The Ihara-Selberg zeta function of a tree lattice," *International Journal of Mathematics*, vol. 6, pp.717-797, 1992.

[42] G. Li, M. Semerci, B. Yener, and M.J. Zaki, "Effective graph classification based on topological and label attributes," *Statistical Analysis and Data Mining*, vol. 5, pp. 265-283, 2012.

[43] A.K. Debnath, R.L. Lopez de Compadre, G. Debnath, A.J. Shusterman, and C. Hansch, Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds, Correlation with molecular orbital energies and hydrophobicity," *J Med Chem*, vol. 34, pp. 786-797, 1991.

[44] I. Schomburg, A. Chang, C. Ebeling, M. Gremse, C. Heldt, G. Huhn, and D. Schomburg, "The enzyme database: Updates and major new developments," *Nucleic Acids Research*, vol. 32, pp. 431-433, 2004.

[45] J.C. Platt, "Fast training of support vector machines using sequential minimal optimization," *Sch ölkopf, B., Burges, C.J.C., and Smola, A.J. (Eds.) Advances in Kernel Methods*, MIT Press, pp. 185-208, 1999.

[46] I.H. Witten, E. Frank, and M.A. Hall, "Data Mining: Practical machine learning tools and techniques," Morgan Kaufmann, 2011.

[47] N. Shervashidze, P. Schweitzer, E.J. Leeuwen, K. Mehlhorn, and K.M. Borgwardt, "Weisfeiler-Lehman graph kernels," *Journal of Machine Learning Research*, vol. 1, pp. 1-48, 2010.

[48] H. Qiu, and Edwin R. Hancock, "Clustering and embedding using commute times," *IEEE Transactions on Pattern Analysis Machine Intellgence*, vol. 29, pp. 1873-1890, 2007.

[49] B. Xiao, Edwin R. Hancock, and Richard C. Wilson, "Graph characteristics from the heat kernel trace," *Pattern Recognition*, vol. 42, pp. 2589-2606, 2009.

[50] T. Gärtner, P.A. Flach, S. Wrobel, "On graph kernels: Hardness results and efficient alternatives," in *Proceedings of Conference on Learning Theory*, 2003, pp. 129-143.

[51] K.M. Borgwardt, and H.P. Kriegel, "Shortest-Path kernels on graphs, in *Proceedings of IEEE International Conference on Data Mining*, 2005, pp. 74-81.

[52] D. Haussler, "Convolution kernels on discrete structures," in *Technical Report UCS-CRL-99-10 of University of California at Santa Cruz*, 1999.

[53] P. Ren, T. Aleksic, D. Emms, R.C. Wilson, and E.R. Hancock, "Quantum walks, Ihara zeta functions and cospectrality in regular graphs," *Quantum Information Processing*, vol.10, pp.405-417, 2011.