MEGA: Multi-View Semi-Supervised Clustering of Hypergraphs

Joyce Jiyoung Whang[*], Rundong Du[†], Sangwon Jung[*], Geon Lee[*], Barry Drake[‡], Qingqing Liu[†], Seonggoo Kang[§], and Haesun Park[†]

[*] Sungkyunkwan University (SKKU), [†] Georgia Institute of Technology, [‡] Georgia Tech Research Institute, [§] Naver Corporation

[*]{jjwhang, s.jung, geonlee}@skku.edu, [†]{rdu, qqliu}@gatech.edu, [†] hpark@cc.gatech.edu, [‡] barry.drake@gtri.gatech.edu, [§] seong.goo.kang@navercorp.com

ABSTRACT

Complex relationships among entities can be modeled very effectively using hypergraphs. Hypergraphs model real-world data by allowing a hyperedge to include two or more entities. Clustering of hypergraphs enables us to group the similar entities together. While most existing algorithms solely consider the connection structure of a hypergraph to solve the clustering problem, we can boost the clustering performance by considering various features associated with the entities as well as auxiliary relationships among the entities. Also, we can further improve the clustering performance if some of the labels are known and we incorporate them into a clustering model. In this paper, we propose a semi-supervised clustering framework for hypergraphs that is able to easily incorporate not only multiple relationships among the entities but also multiple attributes and content of the entities from diverse sources. Furthermore, by showing the close relationship between the hypergraph normalized cut and the weighted kernel K-Means, we also develop an efficient multilevel hypergraph clustering method which provides a good initialization with our semi-supervised multi-view clustering algorithm. Experimental results show that our algorithm is effective in detecting the ground-truth clusters and significantly outperforms other state-of-the-art methods.

PVLDB Reference Format:
Joyce Jiyoung Whang, Rundong Du, Sangwon Jung, Geon Lee, Barry Drake, Qingqing Liu, Seonggoo Kang, and Haesun Park. MEGA: Multi-View Semi-Supervised Clustering of Hypergraphs. *PVLDB*, 13(5): 698-711, 2020.
DOI: https://doi.org/10.14778/3377369.3377378

1. INTRODUCTION

A hypergraph can be considered as a generalization of a graph, which is defined by a set of nodes and a set of hyperedges where a hyperedge connects two or more nodes. Hypergraphs are useful tools to model complex real-world data that include higher order relationships among objects. They have been studied in diverse applications including computer vision [4,28], VLSI CAD [13], bioinformatics [33], and social network analysis [31]. In [4], for example, a hypergraph-based image retrieval and tagging system has been proposed where a hypergraph is utilized to encode additional information which is hard to be represented in a normal graph.

Hypergraph clustering algorithms aim to cluster the nodes based on the connection structure of a hypergraph such that highly connected nodes are assigned to the same cluster. Being different from the traditional graph clustering problem [5,11,36], a hypergraph clustering algorithm should be able to appropriately handle the hyperedges. For example, the hypergraph normalized cut [43] is one of the well-known objectives for hypergraph clustering. We show that the hypergraph normalized cut objective is mathematically equivalent to the weighted kernel K-Means objective. This analysis is an extension of [5] to hypergraphs. Based on the equivalence of the objectives, we develop an efficient multilevel hypergraph clustering algorithm, `hGraclus`, which also seamlessly extends the algorithm in [5] to hypergraphs. This efficient structure-based hypergraph clustering algorithm provides a good initialization of our proposed semi-supervised multi-view model which is described below.

While the connection structure of a hypergraph is a key factor that should be taken into account for hypergraph clustering, we can improve the clustering performance by incorporating various attributes or features of the objects and other multiple auxiliary relationships among the objects if they are available. To incorporate both graph structure and node attributes for the traditional graph clustering, various approaches have been studied including a unified distance-based model [44], model-based methods [30, 38], and a popularity-based conditional link model [39]. These methods can be considered as multi-view clustering methods [3,23] in that they consider different views of the objects, e.g., various attributes of the objects as well as the graph structure, to derive a clustering of the objects.

We note that most existing multi-view clustering methods are based on unsupervised learning. However, if the labels of a subset of the objects are available, we can boost the clustering performance by incorporating these partially available labels into a clustering model [18]. Semi-supervised clustering methods [22,40,41] exploit the partially observed labels of the objects to improve the clustering performance. However, most existing semi-supervised methods do not take into account multi-view clustering, i.e., they just focus on

Proceedings of the VLDB Endowment, Vol. 13, No. 5
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3377369.3377378

a single view of the objects. We further discuss the related work in Section 5.

In this paper, we develop a semi-supervised multi-view clustering algorithm that can effectively handle hypergraphs. We simultaneously consider multi-view clustering and semi-supervised learning to not only reflect multiple views of data (i.e., multiple attributes or content of the objects as well as auxiliary relationships among the objects besides the connection structure of a hypergraph) but also improve the clustering accuracy by utilizing partially available labels of objects. In our method, the hypergraph structure is considered as one of the views of given objects. Even though we start our discussion from hypergraph clustering and some of our analysis is specific to hypergraphs, our method can be considered to be a general framework for semi-supervised multi-view clustering in that the input data is not necessarily restricted to a hypergraph. Also, our method is flexible in the sense that it can be used as either an unsupervised learning method or a semi-supervised method. That is, when there is no prior knowledge about the labels, our method works as an unsupervised multi-view clustering method.

We name our method `MEGA` by abbreviating Multi-view sEmi-supervised hyperGrAph clustering. We show that `MEGA` is efficiently initialized by `hGraclus` which is our multilevel optimization method for the hypergraph normalized cut. We compare the performance of `MEGA` with 13 different baseline methods, and the experimental results show that our method significantly outperforms other state-of-the-art hypergraph clustering methods, multi-view clustering methods, and semi-supervised learning methods in terms of identifying the ground-truth clusters.

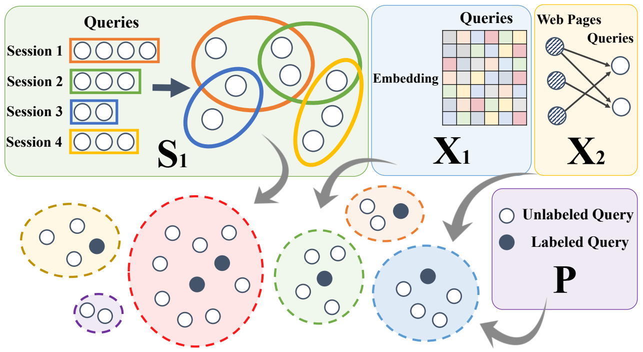
We summarize the main contributions of this paper.

- We show that the hypergraph normalized cut objective is mathematically equivalent to the weighted kernel K-Means objective by defining appropriate kernel and weight matrices (Section 3.1).
- Based on the equivalence of the objectives, we develop an efficient multilevel hypergraph clustering algorithm, `hGraclus`, which optimizes the hypergraph normalized cut in multiple scales using the iterative weighted kernel K-Means algorithm (Section 3.2–3.4).
- Based on the relationship between the hypergraph normalized cut and SymNMF [17] (Section 4.1), we propose a multi-view clustering objective function that can incorporate multiple views of data in addition to the hypergraph structure (Section 4.2).
- We extend our multi-view clustering objective into semi-supervised clustering (Section 4.3).
- We develop a reliable solution procedure to optimize the proposed objective function which is effectively initialized by `hGraclus` (Section 4.4–4.6).
- We show that `MEGA` significantly outperforms 13 different state-of-the-art methods on real-world datasets in terms of the F1, accuracy, and NMI scores (Section 6)[1].

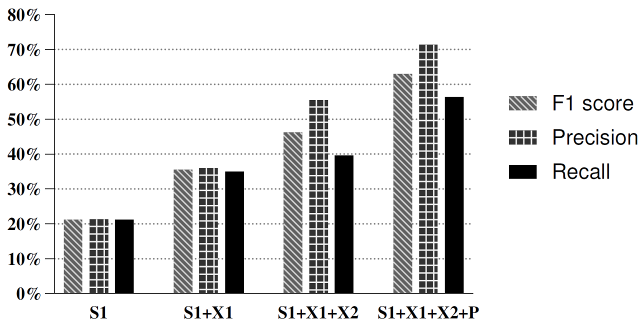
Motivating Example

Let us consider Web query clustering as a real-world application of our model. We get a real-world Web query dataset from NAVER (which is the largest search engine company in South Korea) who also provides the ground-truth clusters. More details about the datasets are described in Section 6.

[1]All the datasets and the codes are available at `http://bigdata.cs.skku.edu`.



(a) Semi-supervised multi-view clustering of Web queries. A query session is modeled as a hypergraph $\boldsymbol{S}_1$. An embedding matrix for the queries $\boldsymbol{X}_1$ and the relationship between the queries and clicked Web pages $\boldsymbol{X}_2$ are also given. A few queries are labeled (denoted by $\boldsymbol{P}$). Our goal is to cluster the queries by incorporating all of this information.



(b) Clustering performance on Web queries according to different numbers of views. As an additional view is incorporated, the clustering performance is improved. Incorporating multiple views as well as partial supervision into a clustering model plays a pivotal role in improving the clustering performance.

Figure 1: Semi-supervised multi-view clustering with a hypergraph

The Web queries can be clustered based on query session information, the semantics of the queries, and click-logs. For Web query clustering, an important feature is a query session that represents a set of Web queries that are made by a user during a short period of time. If a set of queries are entered in one session, we can infer that those queries are closely related with each other. One way to encode this query session information is to use a hypergraph. Let us represent a query as a node and a query session as a hyperedge that connects the queries that appear in the same session. Since more than two queries can appear in one session, we need a hyperedge to appropriately represent a query session. Figure 1(a) shows an example of the hypergraph ($\boldsymbol{S}_1$). Along with the hypergraph, there can be a word embedding matrix for the Web queries ($\boldsymbol{X}_1$: keyword $\times$ query) which encodes the semantics of the queries. Also, a search engine company usually traces clicked Web pages for a given query, which can be represented by a bipartite graph ($\boldsymbol{X}_2$: web pages $\times$ query) where the queries and the clicked Web pages are modeled as two different types of nodes. Furthermore, we assume that the labels of a small subset of the queries are available, and we utilize these partially observed labels which are denoted by $\boldsymbol{P}$ (cluster $\times$ query) in Figure 1(a). Our goal is to appropriately cluster the Web queries by incorporating all of this information.

We investigate whether we can actually improve the clustering performance by adding additional views of objects (besides the hypergraph structure) and incorporating partially observed labels into a clustering model. Figure 1(b) shows the clustering performance of our method when different numbers of views are considered. Since we have the ground-truth clusters for the query dataset, we can compare the algorithmic solutions and the ground-truth clusters. The F1 score is computed by calculating the harmonic mean of the presented precision and recall. Higher F1, precision, and recall scores indicate better clustering performance. In Figure 1(b), the three leftmost bars above $\boldsymbol{S}_1$ indicate the performance of `hGraclus` where only the hypergraph is considered to cluster the Web queries. The three bars above $\boldsymbol{S}_1 + \boldsymbol{X}_1$ indicate the performance of `MEGA` when we consider both the hypergraph $\boldsymbol{S}_1$ and the embedding matrix $\boldsymbol{X}_1$. Similarly, the bars above $\boldsymbol{S}_1 + \boldsymbol{X}_1 + \boldsymbol{X}_2$ indicate the performance of `MEGA` with $\boldsymbol{S}_1$, $\boldsymbol{X}_1$, and $\boldsymbol{X}_2$. The rightmost three bars above $\boldsymbol{S}_1 + \boldsymbol{X}_1 + \boldsymbol{X}_2 + \boldsymbol{P}$ indicate the performance of `MEGA` when we incorporate all of the available views, $\boldsymbol{S}_1$, $\boldsymbol{X}_1$, and $\boldsymbol{X}_2$, and partial supervision $\boldsymbol{P}$. From Figure 1(b), we note that, as we incorporate an additional view of the queries, the clustering performance is improved. Even though it is not guaranteed that the clustering performance always increases according to the number of views in general, our query clustering example shows the benefit of adding a view and partial supervision.

2. HYPERGRAPH CLUSTERING

We briefly review the concept of hypergraphs, and formally state the hypergraph clustering problem.

2.1 Hypergraphs

A hypergraph G is represented by $G = (\mathcal{V}, \mathcal{E}, \mathbf{f})$ where $\mathcal{V}$ is a set of vertices, $\mathcal{E}$ is a set of hyperedges, and $\mathbf{f} \in \mathbb{R}_+^{|\mathcal{E}|}$ is a vector of nonnegative weights for the hyperedges ($\mathbb{R}_+$ denotes nonnegative real numbers). Assume that there are n vertices and m hyperedges, i.e., $|\mathcal{V}| = n$ and $|\mathcal{E}| = m$. A hyperedge $\mathfrak{e}_j \in \mathcal{E}$ is a set that consists of two or more vertices, and $\mathfrak{e}_1 \cup \mathfrak{e}_2 \cup \cdots \cup \mathfrak{e}_m = \mathcal{V}$. A positive weight f_j is associated with a hyperedge $\mathfrak{e}_j$ ($j = 1, \cdots, m$).

We consider an incidence matrix $\boldsymbol{A} \in \{0,1\}^{n \times m}$ of the hypergraph G such that $a_{ij} = 1$ if the i-th vertex is included in the j-th hyperedge and $a_{ij} = 0$ otherwise. Given this notation, the degree of a vertex is defined to be $\mathtt{deg}(v_i) = \sum_j f_j a_{ij}$ and the degree of a hyperedge is defined to be $\mathtt{deg}(\mathfrak{e}_j) = |\mathfrak{e}_j| = \sum_i a_{ij}$. Let $\boldsymbol{D}_v \in \mathbb{R}_+^{n \times n}$ denote the degree diagonal matrix for vertices where the i-th diagonal element corresponds to $\mathtt{deg}(v_i)$. Similarly, let $\boldsymbol{D}_e \in \mathbb{R}_+^{m \times m}$ denote the degree diagonal matrix for hyperedges where the j-th diagonal element corresponds to $\mathtt{deg}(\mathfrak{e}_j)$. We also use $\boldsymbol{F} \in \mathbb{R}_+^{m \times m}$ to represent the weight diagonal matrix for hyperedges where the j-th diagonal element indicates f_j.

2.2 Hypergraph Normalized Cut

Given a hypergraph $G = (\mathcal{V}, \mathcal{E}, \mathbf{f})$, the goal of hypergraph clustering is to partition the vertex set $\mathcal{V}$ into k disjoint clusters while highly connected vertices are assigned to the same cluster. To solve this hypergraph clustering problem, the hypergraph normalized cut objective has been considered [43]. For a cluster $\mathcal{V}_i \subset \mathcal{V}$, the hyperedge boundary $\partial(\mathcal{V}_i)$ of $\mathcal{V}_i$ is defined to be the set of hyperedges that include vertices in both $\mathcal{V}_i$ and ${\mathcal{V}_i}^c$ where ${\mathcal{V}_i}^c = \mathcal{V} \setminus \mathcal{V}_i$. That is, $\partial(\mathcal{V}_i) := \{\mathfrak{e}_j \in \mathcal{E} : \mathfrak{e}_j \cap \mathcal{V}_i \neq \emptyset, \mathfrak{e}_j \cap {\mathcal{V}_i}^c \neq \emptyset\}$. Recall that f_j denotes the weight of $\mathfrak{e}_j$. As suggested in [43], the hyperedge cut of $\mathcal{V}_i$, denoted by $\mathtt{hCut}(\mathcal{V}_i)$, is defined to be

$$\mathtt{hCut}(\mathcal{V}_i) := \sum_{\mathfrak{e}_j \in \partial(\mathcal{V}_i)} f_j \frac{|\mathfrak{e}_j \cap \mathcal{V}_i||\mathfrak{e}_j \cap {\mathcal{V}_i}^c|}{\mathtt{deg}(\mathfrak{e}_j)}. \quad (1)$$

Let us define the volume of $\mathcal{V}_i$ to be the sum of vertex degrees in $\mathcal{V}_i$, i.e., $\mathtt{vol}(\mathcal{V}_i) := \sum_{v \in \mathcal{V}_i} \mathtt{deg}(v)$. Then, the hypergraph normalized cut objective is defined to be

$$\mathtt{hNCut}(G) := \min_{\mathcal{V}_1, \cdots, \mathcal{V}_k} \sum_{i=1}^{k} \frac{\mathtt{hCut}(\mathcal{V}_i)}{\mathtt{vol}(\mathcal{V}_i)}. \quad (2)$$

We can rewrite (2) using matrices and vectors by introducing an indicator matrix $\boldsymbol{Y} \in \{0,1\}^{n \times k}$ where $y_{ij} = 1$ if the i-th node belongs to the j-th cluster and $y_{ij} = 0$ otherwise. Let $\mathbf{y}_j$ denote the j-th column of $\boldsymbol{Y}$. Then, (2) can be expressed as a trace maximization problem as follows:

$$\max_{\widetilde{\boldsymbol{Y}} \geq 0, \widetilde{\boldsymbol{Y}}^T \widetilde{\boldsymbol{Y}} = \boldsymbol{I}_k} \operatorname{trace}(\widetilde{\boldsymbol{Y}}^T {\boldsymbol{D}_v}^{-1/2} \boldsymbol{A} \boldsymbol{F} {\boldsymbol{D}_e}^{-1} \boldsymbol{A}^T {\boldsymbol{D}_v}^{-1/2} \widetilde{\boldsymbol{Y}}) \quad (3)$$

where the j-th column of $\widetilde{\boldsymbol{Y}}$ is defined to be $\dfrac{{\boldsymbol{D}_v}^{1/2} \mathbf{y}_j}{\sqrt{{\mathbf{y}_j}^T \boldsymbol{D}_v \mathbf{y}_j}}$. Note that $\widetilde{\boldsymbol{Y}}^T \widetilde{\boldsymbol{Y}} = \boldsymbol{I}_k$. The problem (3) can be solved by spectral clustering [24] which keeps the orthogonality but in general ignores the nonnegativity constraint [43] or a symmetric nonnegative matrix factorization (SymNMF) [17] which keeps the nonnegativity constraint only.

3. HYPERGRAPH NORMALIZED CUT & WEIGHTED KERNEL K-MEANS

We show that the hypergraph normalized cut objective becomes equivalent to the weighted kernel K-means objective when we define appropriate kernel and weights. This equivalence allows us to develop an efficient multilevel hypergraph clustering algorithm which optimizes the hypergraph normalized cut using the iterative weighted kernel K-means algorithm. Our analysis and algorithm presented in this section are extensions of [5] to hypergraphs.

3.1 Equivalence of the Objectives

Given a set of data points $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$, the weighted kernel K-Means clustering objective [5] may be written as

$$\min_{\boldsymbol{U} \in \{0,1\}^{n \times k}} \sum_{j=1}^{k} \sum_{i=1}^{n} u_{ij} \pi_i \|\phi(\mathbf{x}_i) - \mathbf{m}_j\|^2, \mathbf{m}_j = \frac{\sum_{i=1}^{n} u_{ij} \pi_i \phi(\mathbf{x}_i)}{\sum_{i=1}^{n} u_{ij} \pi_i} \quad (4)$$

where π_i indicates a nonnegative weight of $\mathbf{x}_i$, $\phi(\mathbf{x}_i)$ indicates a nonlinear mapping of $\mathbf{x}_i$, and $\boldsymbol{U} \in \{0,1\}^{n \times k}$ is an assignment matrix where $u_{ij} = 1$ if the i-th data point belongs to the j-th cluster and $u_{ij} = 0$ otherwise. Let $\mathbf{u}_j$ denote the j-th column of $\boldsymbol{U}$. By introducing a kernel matrix $\boldsymbol{K} \in \mathbb{R}^{n \times n}$ where $k_{ij} = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$, and the weight diagonal matrix $\boldsymbol{\Pi} \in \mathbb{R}^{n \times n}$ where the i-th diagonal entry indicates π_i, we can rewrite (4) in the form of the trace maximization problem

$$\max_{\widetilde{\boldsymbol{U}} \geq 0, \widetilde{\boldsymbol{U}}^T \widetilde{\boldsymbol{U}} = \boldsymbol{I}_k} \operatorname{trace}(\widetilde{\boldsymbol{U}}^T \boldsymbol{\Pi}^{1/2} \boldsymbol{K} \boldsymbol{\Pi}^{1/2} \widetilde{\boldsymbol{U}}) \quad (5)$$

where the j-th column of $\widetilde{\boldsymbol{U}}$ is defined to be $\dfrac{\boldsymbol{\Pi}^{1/2} \mathbf{u}_j}{\sqrt{{\mathbf{u}_j}^T \boldsymbol{\Pi} \mathbf{u}_j}}$.

Let us compare (3) and (5). First, we notice that the assignment matrices $\widetilde{\boldsymbol{Y}}$ in (3) and $\widetilde{\boldsymbol{U}}$ in (5) are defined in the same way. Now, in (5), let us define the weight and kernel matrices to be $\boldsymbol{\Pi} := \boldsymbol{D}_v$ and $\boldsymbol{K} := \boldsymbol{D}_v{}^{-1}\boldsymbol{A}\boldsymbol{F}\boldsymbol{D}_e{}^{-1}\boldsymbol{A}^T\boldsymbol{D}_v{}^{-1}$ (Note that $\boldsymbol{K}$ is a positive semidefinite matrix since $\boldsymbol{K} = \boldsymbol{M}\boldsymbol{M}^T$ where $\boldsymbol{M} = \boldsymbol{D}_v{}^{-1}\boldsymbol{A}\boldsymbol{F}^{1/2}\boldsymbol{D}_e{}^{-1/2}$). Then, we see that (5) becomes equivalent to (3). Thus, the hypergraph normalized cut objective (3) is mathematically equivalent to the weighted kernel K-Means objective (5).

3.2 Optimizing the Hypergraph Normalized Cut using the Weighted Kernel K-Means

The weighted kernel K-Means objective can be optimized by the simple iterative algorithm [5]. Since the hypergraph normalized cut objective can be converted into the weighted kernel K-Means objective as discussed in Section 3.1, we can optimize the hypergraph normalized cut by the weighted kernel K-Means algorithm with appropriate kernel and weights.

In the weighted kernel K-Means algorithm, we first initialize the clusters, and set the tolerance value ϵ and the maximum number of iterations t_{max}. For every vertex $v_i \in \mathcal{V}$, we compute the distance between the vertex v_i and the k clusters, and assign the vertex to its closest cluster. We repeat this process until the change in the objective value becomes smaller than ϵ or the number of iterations reaches t_{max}. A key of this process is how to quantify the distance $\texttt{dist}(v_i, \mathcal{V}_c)$ between a vertex v_i and a cluster $\mathcal{V}_c$. As discussed in Section 3.1, by defining the weight and kernel matrices to be $\boldsymbol{\Pi} := \boldsymbol{D}_v$ and $\boldsymbol{K} := \boldsymbol{D}_v{}^{-1}\widehat{\boldsymbol{A}}\boldsymbol{D}_v{}^{-1}$ where $\widehat{\boldsymbol{A}} := \boldsymbol{A}\boldsymbol{F}\boldsymbol{D}_e{}^{-1}\boldsymbol{A}^T$, the distance $\texttt{dist}(v_i, \mathcal{V}_c)$ is computed by

$$\texttt{dist}(v_i, \mathcal{V}_c) = \frac{\sum_{v_j \in \mathcal{V}_c, v_l \in \mathcal{V}_c} \hat{a}_{jl}}{\texttt{vol}(\mathcal{V}_c)^2} - \frac{2\sum_{v_j \in \mathcal{V}_c} \hat{a}_{ij}}{\texttt{deg}(v_i)\,\texttt{vol}(\mathcal{V}_c)} \qquad (6)$$

where $\hat{a}_{ij}$ denotes the i-th row and the j-th column entry in $\widehat{\boldsymbol{A}}$. Indeed, $\hat{a}_{ij}$ can be represented by $\hat{a}_{ij} = \sum_{\mathfrak{e} \in \mathcal{Q}_e} \frac{f(\mathfrak{e})}{\texttt{deg}(\mathfrak{e})}$ where $\mathcal{Q}_e = \{\mathfrak{e} \in \mathcal{E} : v_i \in \mathfrak{e}, v_j \in \mathfrak{e}\}$ and $f(\mathfrak{e})$ denotes the weight of the hyperedge $\mathfrak{e}$. This implies that we construct $\widehat{\boldsymbol{A}}$ which encodes the nodes × nodes relationship where each component $\hat{a}_{ij}$ incorporates the hyperedges that are shared by the vertices v_i and v_j with appropriate normalization. This interpretation allows us to further extend our idea to clustering of hyperedges. To cluster the hyperedges in a hypergraph, we can intuitively construct $\boldsymbol{\Pi} := \boldsymbol{D}_e$ and $\boldsymbol{K} := \boldsymbol{D}_e{}^{-1}\widetilde{\boldsymbol{A}}\boldsymbol{D}_e{}^{-1}$ where $\widetilde{\boldsymbol{A}} := \boldsymbol{A}^T\boldsymbol{D}_v{}^{-1}\boldsymbol{A}$ to encode the hyperedges × hyperedges relationship. Each component $\tilde{a}_{ij}$ in $\widetilde{\boldsymbol{A}}$ incorporates the vertices that are shared by the hyperedges $\mathfrak{e}_i$ and $\mathfrak{e}_j$. Specifically, $\tilde{a}_{ij} = \sum_{v \in \mathcal{Q}_v} \frac{1}{\texttt{deg}(v)}$ where $\mathcal{Q}_v = \{v \in \mathcal{V} : v \in \mathfrak{e}_i, v \in \mathfrak{e}_j\}$.

3.3 Multilevel Hypergraph Clustering

A multilevel graph clustering framework has been known to be an efficient way to solve large-scale graph clustering problems as noted in [12], [5], and [35]. However, very few studies have been conducted on hypergraph clustering, especially from the hypergraph normalized cut optimization perspective. We develop a multilevel hypergraph clustering algorithm, **hGraclus**, which refines clustering in multiple scales using the weighted kernel K-Means algorithm. By refining the clustering results in multiple scales, we can get qualitatively better solutions than those achieved by a single-level optimization. As described in Algorithm 1, **hGraclus** consists of three phases: coarsening phase, base clustering phase, and refinement phase. We present the clustering performance of **hGraclus** in Section 3.4.

Algorithm 1: Multilevel Hypergraph Clustering via Weighted Kernel K-Means (hGraclus)

```
Input: $G = (\mathcal{V}, \mathcal{E}, \mathbf{f})$, $k$
Output: $\mathcal{V}_1, \mathcal{V}_2, \cdots, \mathcal{V}_k$
 1: /* Coarsening Phase */
 2: Set $t = 0$, $G^{(0)} = G$, $\mathcal{V}^{(0)} \leftarrow \mathcal{V}$, $\mathcal{E}^{(0)} \leftarrow \mathcal{E}$, $\mathbf{f}^{(0)} \leftarrow \mathbf{f}$.
 3: repeat
 4:     $t \leftarrow t + 1$, $l \leftarrow t$.
 5:     Create $G^{(t)}$ from $G^{(t-1)}$ by merging pairs of vertices
        that share a hyperedge. The vertices that have a high
        normalized hyperedge cut defined in (7) are merged.
 6: until $|\mathcal{V}^{(t)}| \geq 0.1 \times |\mathcal{V}^{(0)}|$
 7: /* Base Clustering Phase */
 8: Partition $G^{(l)}$ into $k$ clusters by a base clustering scheme.
 9: /* Refinement Phase */
10: for $t = l, l-1, \cdots, 1$ do
11:     Initialize a clustering of $G^{(t-1)}$ by projecting the clus-
        tering result of $G^{(t)}$ into $G^{(t-1)}$.
12:     Refine the clustering of $G^{(t-1)}$ by running the
        weighted kernel K-Means algorithm as described in
        Section 3.2 where the distance $\texttt{dist}(v_i, \mathcal{V}_c)$ between a
        vertex $v_i$ and a cluster $\mathcal{V}_c$ is computed by (6).
13: end for
```

3.3.1 *Coarsening Phase*

Given the original hypergraph $G^{(0)}$, we create a series of smaller hypergraphs by merging vertices. That is, we create $G^{(t)}$ from $G^{(t-1)}$ such that the number of vertices in $G^{(t)}$ is smaller than that in $G^{(t-1)}$. To determine which vertices should be merged, we consider pairs of vertices that share a hyperedge. Let us consider v and v' in $G^{(t-1)}$, and assume that v and v' share a set of hyperedges represented by $\mathcal{Q}_e = \{\mathfrak{e} \in \mathcal{E}^{(t-1)} : v \in \mathfrak{e}, v' \in \mathfrak{e}\}$ where $\mathcal{E}^{(t-1)}$ denotes the set of hyperedges of $G^{(t-1)}$. If we consider v and v' to be individual clusters, the normalized hyperedge cut between v and v' can be represented by

$$\texttt{hNCut}(v, v') = \frac{\sum_{\mathfrak{e} \in \mathcal{Q}_e} \frac{f(\mathfrak{e})}{\texttt{deg}(\mathfrak{e})}}{\texttt{deg}(v)} + \frac{\sum_{\mathfrak{e} \in \mathcal{Q}_e} \frac{f(\mathfrak{e})}{\texttt{deg}(\mathfrak{e})}}{\texttt{deg}(v')} \qquad (7)$$

where $f(\mathfrak{e})$ denotes the weight of the hyperedge $\mathfrak{e}$ (note that $\texttt{deg}(v)$ and $\texttt{deg}(\mathfrak{e})$ are defined in Section 2.1). In the coarsening phase, our goal is to gradually scale down the input hypergraph while minimizing the normalized hyperedge cut. To achieve this goal, we create $G^{(t)}$ from $G^{(t-1)}$ as follows. Initially, all the vertices in $G^{(t-1)}$ are unmarked. We visit each unmarked vertex v and merge it with an unmarked v' that maximizes (7) to hide the large hyperedge cut. Then, v and v' are marked. If all neighbors of v have been marked, we simply mark v. Once all the vertices are marked, the coarsening from $G^{(t-1)}$ to $G^{(t)}$ is complete. In this way, we repeatedly coarsen the hypergraph until the finally coarsened hypergraph contains less than 10% of the number of vertices in the original hypergraph. Although different kinds of stopping criteria or heuristics can be applied in this coarsening phase, we observe that the performance of **hGraclus** is not largely affected by those heuristics.

Table 1: Clustering performance of `SWS` [28], `SPC` [43], `hMetis` [13] and `hGraclus`. In terms of the hypergraph normalized cut (`hNCut` defined in Section 2.2) and run time (in seconds), `hGraclus` shows the best performance.

		`SWS`	`SPC`	`hMetis`	`hGraclus`
QUERY	`hNCut`	1.276	3.286	0.659	**0.550**
	Run Time	51.3	0.131	1.230	**0.005**
GENE	`hNCut`	0.720	2.361	0.512	**0.496**
	Run Time	193.5	0.267	0.519	**0.009**
CORA	`hNCut`	2.163	4.542	0.588	**0.512**
	Run Time	871.7	0.090	0.432	**0.008**
DBLP5	`hNCut`	0.937	2.920	0.206	**0.131**
	Run Time	2331.0	4.628	1.387	**0.057**
DBLP10	`hNCut`	2.149	6.289	0.435	**0.321**
	Run Time	8068.3	20.7	3.394	**0.114**

3.3.2 *Base Clustering Phase*

Once the original hypergraph is transformed into a small hypergraph, we apply the base clustering scheme by adopting similar approaches used in [5]. The idea is to partition the small hypergraph into k clusters by running a traditional graph clustering method on $\widehat{\boldsymbol{A}}$ (or $\widetilde{\boldsymbol{A}}$ for hyperedge clustering) discussed in Section 3.2.

3.3.3 *Refinement Phase*

We first initialize a clustering of $G^{(t-1)}$ by projecting the clustering result of $G^{(t)}$ into $G^{(t-1)}$ (projection step). That is, if a supernode v_i in $G^{(t)}$ belongs to cluster $\mathcal{V}_c$, then we initially assign the nodes in $G^{(t-1)}$ which were merged by the supernode v_i to $\mathcal{V}_c$. Then, we run the weighted kernel K-Means algorithm to refine the clustering of $G^{(t-1)}$ (refinement step). Note that the distance between a vertex and a cluster is computed by (6) as described in Section 3.2. Since the projection step provides $G^{(t-1)}$ with a good initialization, the weighted kernel K-Means algorithm usually converges in a small number of iterations. This refinement step plays the most crucial role in optimizing the hypergraph normalized cut. From the smallest coarsened hypergraph to the original hypergraph, we refine the clustering results l times at different scales. This multi-level refinement enables us to achieve qualitatively good results in practice since we apply the weighted kernel K-Means algorithm multiple times on the given hypergraph with different scales.

3.4 Clustering Performance of `hGraclus`

We compare the clustering performance of `hGraclus` (Algorithm 1) with those of `hMetis` [13], a spectral hypergraph partitioning method [43] (denoted by `SPC`) , and the `SWS` [28] method. These four methods cluster a hypergraph based solely on the hypergraph structure. While both `hGraclus` and `hMetis` are multilevel hypergraph clustering algorithms, `hMetis` is designed to produce almost equal sized clusters while `hGraclus` does not have explicit constraints on cluster sizes. Table 1 shows the clustering performance of the methods on the five datasets described in Section 6.1. We compare the hypergraph normalized cut, i.e., `hNCut` defined by (2) in Section 2.2, and the run time (in seconds). A lower hypergraph normalized cut value indicates a better clustering. We see that `hGraclus` is the fastest method and achieves the smallest hypergraph normalized cut on all the datasets.

4. A SEMI-SUPERVISED LEARNING FRAMEWORK FOR MULTI-VIEW CLUSTERING OF HYPERGRAPHS

We extend our idea to a semi-supervised clustering method which incorporates not only the hypergraph structure but also various features associated with the objects as well as multiple auxiliary relationships among the objects.

4.1 Hypergraph Normalized Cut & SymNMF

There is a close relationship between the hypergraph normalized cut objective and SymNMF [17]. Given a nonnegative symmetric matrix $\boldsymbol{B} \in \mathbb{R}_+^{n\times n}$ and a reduced rank k, the SymNMF of $\boldsymbol{B}$ is defined to be the problem of finding a nonnegative matrix $\boldsymbol{V} \in \mathbb{R}_+^{n\times k}$ such that the difference between $\boldsymbol{B}$ and $\boldsymbol{V}\boldsymbol{V}^T$ is minimized. That is,

$$\min_{\boldsymbol{V}\geq 0} \|\boldsymbol{B} - \boldsymbol{V}\boldsymbol{V}^T\|_F^2. \tag{8}$$

Recall the hypergraph normalized cut defined in (3). Let $\boldsymbol{B} := \boldsymbol{D}_v{}^{-1/2}\boldsymbol{A}\boldsymbol{F}\boldsymbol{D}_e{}^{-1}\boldsymbol{A}^T\boldsymbol{D}_v{}^{-1/2}$. Then, by following the similar analysis suggested in [16, 17], we can rewrite (3) as

$$\max_{\widetilde{\boldsymbol{Y}}\geq 0, \widetilde{\boldsymbol{Y}}^T\widetilde{\boldsymbol{Y}}=\boldsymbol{I}_k} \text{trace}(\widetilde{\boldsymbol{Y}}^T\boldsymbol{B}\widetilde{\boldsymbol{Y}}) \equiv \min_{\widetilde{\boldsymbol{Y}}\geq 0, \widetilde{\boldsymbol{Y}}^T\widetilde{\boldsymbol{Y}}=\boldsymbol{I}_k} \|\boldsymbol{B} - \widetilde{\boldsymbol{Y}}\widetilde{\boldsymbol{Y}}^T\|_F^2. \tag{9}$$

Note that in (9) and (8), although the constraints on $\widetilde{\boldsymbol{Y}}$ and $\boldsymbol{V}$ are different, the function to minimize is the same. Therefore, the hypergraph normalized cut can be reformulated as a SymNMF problem. While the method discussed in Section 3 directly optimizes the hypergraph normalized cut, we now consider solving the SymNMF problem because reformulating the problem in this form is beneficial to extending our model to multi-view and semi-supervised clustering settings which will be discussed in the following sections. While SymNMF [17] is an unsupervised graph clustering method, our proposed model is a multi-view semi-supervised clustering method that can also handle hypergraphs.

4.2 Multi-View Clustering of Hypergraphs

If our goal is to cluster a hypergraph based solely on the structure, we can consider optimizing the hypergraph normalized cut using the `hGraclus` algorithm proposed in Section 3.3. However, along with a hypergraph, we often have multiple features or attributes for the objects, multiple auxiliary relationships among the objects or the similarity information between the objects. Now, we discuss a way to incorporate all this information into the hypergraph clustering problem so that we get a multi-view clustering model.

Let $\boldsymbol{X}_i \in \mathbb{R}_+^{l_i\times n}$, $i = 1, 2, \cdots, p$, denote the i-th feature set of the n objects (i.e., the i-th view) where each object is represented by l_i nonnegative features and p is the number of available feature sets. We may also have multiple relationships or similarity matrices for the objects. Let $\boldsymbol{S}_j \in \mathbb{R}_+^{n\times n}$, $j = 1, 2, \cdots, q$, denote a symmetric matrix that represents a relationship between the objects, where q is the number of different relationships or similarity matrices. In this setting, the hypergraph structure can be represented by $\boldsymbol{S}_j$ where $\boldsymbol{S}_j := \boldsymbol{D}_v{}^{-1/2}\boldsymbol{A}\boldsymbol{F}\boldsymbol{D}_e{}^{-1}\boldsymbol{A}^T\boldsymbol{D}_v{}^{-1/2}$ due to the analysis in Section 4.1. We develop a unified framework to incorporate all the information provided by $\boldsymbol{X}_i$ and $\boldsymbol{S}_j$ into the clustering using a joint nonnegative matrix factorization (NMF) approach. The key idea is that each of the given matrices,

$\boldsymbol{X}_i$ and $\boldsymbol{S}_j$, can be simultaneously approximated by a product of two nonnegative low-rank matrices where one of the factors is shared across the given matrices and regarded as the cluster assignment matrix.

We can approximate $\boldsymbol{X}_i$ by the product $\boldsymbol{W}_i\boldsymbol{H}$ where $\boldsymbol{H} \in \mathbb{R}_+^{k\times n}$ is the common factor (i.e., the shared factor) and $\boldsymbol{W}_i \in \mathbb{R}_+^{l_i\times k}$ is the $\boldsymbol{X}_i$-dependent factor. Then, each column of $\boldsymbol{H}$ can be interpreted as a soft clustering assignment of the corresponding object while each column of $\boldsymbol{W}_i$ is considered as a representative of the corresponding cluster. The index of the maximum coordinate in $\boldsymbol{H}(:,j)$, say i^*, indicates that the j-th object is best represented by the i^*-th column in $\boldsymbol{W}_i$. Therefore, it is reasonable to assign the j-th object to the i^*-th cluster. Each column of $\boldsymbol{H}$ can be interpreted in this way, and thus $\boldsymbol{H}$ can be regarded as a cluster assignment matrix.

When we approximate $\boldsymbol{S}_j$ by a product of two low-rank matrices, we can use the SymNMF [17]. Although a standard symmetric NMF assumes $\boldsymbol{S}_j \approx \boldsymbol{H}^T\boldsymbol{H}$, we let $\boldsymbol{S}_j \approx \widehat{\boldsymbol{H}}_j{}^T\boldsymbol{H}$ while we impose $\widehat{\boldsymbol{H}}_j \approx \boldsymbol{H}$ to solve the problem more efficiently [6]. By setting $\boldsymbol{S}_j := \boldsymbol{D}_v{}^{-1/2}\boldsymbol{AFD}_e{}^{-1}\boldsymbol{A}^T\boldsymbol{D}_v{}^{-1/2}$, we optimize the hypergraph normalized cut as shown in Section 4.1, and thus we naturally incorporate the hypergraph structure into our multi-view clustering problem. Finally, integrating all of the above, we propose the following objective function:

$$\min_{(\boldsymbol{W}_i,\boldsymbol{H},\widehat{\boldsymbol{H}}_j)\geq 0} \sum_{i=1}^{p} \alpha_i \|\boldsymbol{X}_i - \boldsymbol{W}_i\boldsymbol{H}\|_F^2 + \sum_{j=1}^{q}\beta_j \left\|\boldsymbol{S}_j - \widehat{\boldsymbol{H}}_j{}^T\boldsymbol{H}\right\|_F^2 + \sum_{j=1}^{q}\gamma_j\left\|\widehat{\boldsymbol{H}}_j - \boldsymbol{H}\right\|_F^2 \quad (10)$$

where the parameters α_i and β_j weigh the relative importance of the corresponding terms, and theoretically we need a large value for γ_j to let $\widehat{\boldsymbol{H}}_j$ and $\boldsymbol{H}$ become close. We can rewrite (10) as

$$\min_{(\widetilde{\boldsymbol{W}},\boldsymbol{H},\widehat{\boldsymbol{H}}_j)\geq 0} \left\|\widetilde{\boldsymbol{X}} - \widetilde{\boldsymbol{W}}\boldsymbol{H}\right\|_F^2 + \sum_{j=1}^{q}\beta_j\left\|\boldsymbol{S}_j - \widehat{\boldsymbol{H}}_j{}^T\boldsymbol{H}\right\|_F^2 + \sum_{j=1}^{q}\gamma_j\left\|\widehat{\boldsymbol{H}}_j - \boldsymbol{H}\right\|_F^2 \quad (11)$$

where $\widetilde{\boldsymbol{X}} = \left[\sqrt{\alpha_1}\boldsymbol{X}_1; \sqrt{\alpha_2}\boldsymbol{X}_2; \cdots; \sqrt{\alpha_p}\boldsymbol{X}_p\right]$ and $\widetilde{\boldsymbol{W}} = \left[\sqrt{\alpha_1}\boldsymbol{W}_1; \sqrt{\alpha_2}\boldsymbol{W}_2; \cdots; \sqrt{\alpha_p}\boldsymbol{W}_p\right]$. The $\boldsymbol{H}$ factor is common for all the terms in (11), and plays the role of the cluster assignment matrix that incorporates all the signals captured by $\boldsymbol{X}_i$ and $\boldsymbol{S}_j$. Therefore, by solving (11), we get a solution of the multi-view clustering problem where the multiple views are provided in the form of $\boldsymbol{X}_i$ and $\boldsymbol{S}_j$.

The approximation error of $\min_{(\boldsymbol{W}_i,\boldsymbol{H}\geq 0)} \|\boldsymbol{X}_i - \boldsymbol{W}_i\boldsymbol{H}\|_F^2$ in (10) is bounded below by the optimal NMF residual of $\min_{(\boldsymbol{D},\boldsymbol{E}\geq 0)} \|\boldsymbol{X}_i - \boldsymbol{DE}\|_F^2$. The difference is that $\boldsymbol{H}$ is common in all terms in (10), so more restricted. But $\boldsymbol{D}$ and $\boldsymbol{E}$ in the second equation depend only on $\boldsymbol{X}_i$. Then, the second equation is bounded below by $\min\|\boldsymbol{X}_i - \boldsymbol{DE}\|_F^2$ where low rank factors have no restriction. Therefore, the optimal solution $\boldsymbol{DE}$ here will be the same as the $\bar{\boldsymbol{U}}_{k'}\bar{\boldsymbol{S}}_{k'}\bar{\boldsymbol{V}}_{k'}^T$ where $\boldsymbol{X}_i = \bar{\boldsymbol{U}}\bar{\boldsymbol{S}}\bar{\boldsymbol{V}}^T$ is the SVD of $\boldsymbol{X}_i$ and $\bar{\boldsymbol{U}}_{k'}$ and $\bar{\boldsymbol{V}}_{k'}$ are the first k' columns of $\bar{\boldsymbol{U}}$ and $\bar{\boldsymbol{V}}$, respectively, and $\bar{\boldsymbol{S}}_{k'}$ is the leading principal $k'\times k'$ submatrix of $\bar{\boldsymbol{S}}$. So $\min\|\boldsymbol{X}_i - \boldsymbol{DE}\|_F^2$ is bounded below by the sum of σ_j^2 for $j = k'+1, \cdots, p'$, where $p' = min(l_i, n)$ assuming $\boldsymbol{X}_i \in \mathbb{R}_+^{l_i\times n}$, and σ_j is the j-th singular value of $\boldsymbol{X}_i$.

4.3 Semi-Supervised Learning

While most existing clustering methods have been studied in the context of unsupervised learning, we can improve the clustering performance if we can exploit the information of any available labels of the objects. We now discuss a way to extend our multi-view clustering model defined in (11) to a semi-supervised learning framework. Suppose that we partially observe the labels of the objects. Let $\boldsymbol{P} \in \{0,1\}^{k\times n}$ denote the partially observed labels, i.e., $p_{ij} = 1$ if the j-th object belongs to the i-th cluster. It is important to note that $p_{ij} = 0$ can be interpreted in two different ways: (i) the j-th object does not belong to the i-th cluster, (ii) there is no information about whether the j-th object belongs to the i-th cluster or not. Note that (i) can happen when we observe the label of the j-th object whereas (ii) can happen for the objects whose labels are not observed. To incorporate $\boldsymbol{P}$ in our model (11), we assume that $\boldsymbol{P}$ is approximated by $\boldsymbol{WH}$ where $\boldsymbol{W} \in \mathbb{R}_+^{k\times k}$ and using the same $\boldsymbol{H}$ as in (11). We note that, different from $\boldsymbol{X}_i$ and $\boldsymbol{S}_j$ discussed in Section 4.2, $\boldsymbol{P}$ encodes partially observed information. Thus, we approximate p_{ij} by $\mathbf{w}_i{}^T\mathbf{h}_j$ only if p_{ij} is an observed entry where $\mathbf{w}_i{}^T$ denotes the i-th row of $\boldsymbol{W}$ and $\mathbf{h}_j$ denotes the j-th column of $\boldsymbol{H}$. To encode whether an entry in $\boldsymbol{P}$ is observed or not, we consider entry-wise operations. Following these ideas, we extend (11) to

$$\min_{(\widetilde{\boldsymbol{W}},\boldsymbol{W},\boldsymbol{H},\widehat{\boldsymbol{H}}_j)\geq 0} \left\|\widetilde{\boldsymbol{X}} - \widetilde{\boldsymbol{W}}\boldsymbol{H}\right\|_F^2 + \sum_{j=1}^{q}\beta_j\left\|\boldsymbol{S}_j - \widehat{\boldsymbol{H}}_j{}^T\boldsymbol{H}\right\|_F^2 + \sum_{j=1}^{q}\gamma_j\left\|\widehat{\boldsymbol{H}}_j - \boldsymbol{H}\right\|_F^2 + \|\mathbb{M}\circ(\boldsymbol{P} - \boldsymbol{WH})\|_F^2 \quad (12)$$

where $\mathbb{M} \in \{0,1\}^{k\times n}$ indicates a masking matrix where $m_{ij} = 1$ if p_{ij} is observed and $m_{ij} = 0$ otherwise, and $\circ$ denotes a Hadamard product of the matrices. Note that the factor $\boldsymbol{H}$ is common for all the terms. By optimizing (12) by the algorithm described in Section 4.4, we can get the cluster assignment factor $\boldsymbol{H}$ that incorporates all the signals captured by $\boldsymbol{X}_i$, $\boldsymbol{S}_j$, and $\boldsymbol{P}$.

When we incorporate partial supervision into our model for semi-supervised learning, we can also consider a set of pairwise constraints between objects [1] instead of assuming that we directly observe the labels. Let $\boldsymbol{S} \in \{0,1\}^{n\times n}$ represent the pairwise constraints where $s_{ij} = 1$ indicates the objects i and j belong to the same cluster (i.e., a must-link constraint) whereas $s_{ij} = 0$ indicates they do not belong to the same cluster (i.e., a cannot-link constraint) if s_{ij} is observed. If s_{ij} is not observed, s_{ij} remains to be zero. To distinguish whether $s_{ij} = 0$ indicates a cannot-link constraint or s_{ij} is an unobserved entry, we use a masking matrix $\mathbb{Z} \in \{0,1\}^{n\times n}$ where $z_{ij} = 1$ if s_{ij} is observed and $z_{ij} = 0$ otherwise. In what follows, we formulate this idea.

$$\min_{(\widetilde{\boldsymbol{W}},\boldsymbol{H},\widehat{\boldsymbol{H}}_j,\overline{\boldsymbol{H}})\geq 0} \left\|\widetilde{\boldsymbol{X}} - \widetilde{\boldsymbol{W}}\boldsymbol{H}\right\|_F^2 + \sum_{j=1}^{q}\beta_j\left\|\boldsymbol{S}_j - \widehat{\boldsymbol{H}}_j{}^T\boldsymbol{H}\right\|_F^2 + \sum_{j=1}^{q}\gamma_j\left\|\widehat{\boldsymbol{H}}_j - \boldsymbol{H}\right\|_F^2 + \left\|\mathbb{Z}\circ(\boldsymbol{S} - \overline{\boldsymbol{H}}^T\boldsymbol{H})\right\|_F^2 + \mu\left\|\overline{\boldsymbol{H}} - \boldsymbol{H}\right\|_F^2 \quad (13)$$

where theoretically we need a large value for μ to let the $\overline{\boldsymbol{H}}$ and $\boldsymbol{H}$ become close. In this way, we incorporate partial supervision into our clustering model.

Our proposed clustering objective functions (12) and (13) are able to not only capture diverse signals represented by the hypergraph structure, various attributes of the objects, and multiple auxiliary relationships between the objects but also utilize partially observed labels of the given objects.

Normalization and Parameter Selection

For a better performance of SymNMF as a clustering method, we use the following scaling: Each column of $\boldsymbol{X}_i$ is normalized to have unit L2 norm, which leads to $\|\boldsymbol{X}_i\|_F = \sqrt{n}$, and we scale $\boldsymbol{S}_j$ by $\boldsymbol{D}_j{}^{-1/2}\boldsymbol{S}_j\boldsymbol{D}_j{}^{-1/2}$ where a diagonal element of a diagonal matrix $\boldsymbol{D}_j$ is the sum of the elements on the corresponding row of $\boldsymbol{S}_j$ [17]. To balance the terms $\boldsymbol{X}_i$ and $\boldsymbol{S}_j$, we multiply $\sqrt{n}/\|\boldsymbol{S}_j\|_F$ to the $\boldsymbol{S}_j$ terms. For the terms that involve $\mathbb{M}$ or $\mathbb{Z}$, we normalize each column of $\mathbb{M} \circ \boldsymbol{P}$ to have unit L2 norm, and perform the degree normalization on $\mathbb{Z} \circ \boldsymbol{S}$. When we compare the results of `MEGA` with and without this kind of normalization, we observe that the proper normalization leads to improving the quality of clusters. In particular, the average gain of the normalization is 7.3% in terms of matching to the ground-truth clusters.

In (12) and (13), we set $\gamma_j = \beta_j \max(\boldsymbol{S}_j)$ where $\max(\boldsymbol{S}_j)$ indicates the largest component in $\boldsymbol{S}_j$ to impose a large weight on the term $\left\|\widehat{\boldsymbol{H}}_j - \boldsymbol{H}\right\|_F^2$ to force $\widehat{\boldsymbol{H}}_j$ and $\boldsymbol{H}$ to be very close. Similarly, we set $\mu = \max(\mathbb{Z} \circ \boldsymbol{S})$.

In our experiments, we set $\alpha_i = 1$ for all $i = 1, \cdots p$, and $\beta_j = 1$ for all $j = 1, \cdots, q$. In general, if any prior knowledge about the importance of each view is provided, the α_i and β_j values can be appropriately specified. Otherwise, we can try several different values for α_i and β_j, and choose the parameters that are associated with the best clustering performance on the small training set which is provided for the semi-supervision. Another way to set the parameters is to select the parameters which lead to the smallest normalized objective function value.

4.4 Optimization Algorithm (`MEGA`)

We solve (12) and (13) using an alternating minimization scheme of block coordinate descent (BCD). Taking (12) as an example and assuming $p = 2, q = 1$, then (12) becomes

$$\min_{(\boldsymbol{W}_1,\boldsymbol{W}_2,\widehat{\boldsymbol{H}}_1,\boldsymbol{W},\boldsymbol{H})\geq 0} \alpha_1 \|\boldsymbol{X}_1 - \boldsymbol{W}_1\boldsymbol{H}\|_F^2 + \alpha_2 \|\boldsymbol{X}_2 - \boldsymbol{W}_2\boldsymbol{H}\|_F^2$$
$$+\beta_1 \left\|\boldsymbol{S}_1 - \widehat{\boldsymbol{H}}_1{}^T\boldsymbol{H}\right\|_F^2 + \gamma_1 \left\|\widehat{\boldsymbol{H}}_1 - \boldsymbol{H}\right\|_F^2 + \|\mathbb{M} \circ (\boldsymbol{P} - \boldsymbol{W}\boldsymbol{H})\|_F^2 . \tag{14}$$

We optimize (14) using a 4-block coordinate descent where each sub-problem is defined as follows.

$$\min_{\widetilde{\boldsymbol{W}}\geq 0} \left\|\boldsymbol{H}^T\widetilde{\boldsymbol{W}}^T - \widetilde{\boldsymbol{X}}^T\right\|_F^2 \tag{15}$$

where $\widetilde{\boldsymbol{W}}^T = [\sqrt{\alpha_1}\boldsymbol{W}_1^T \sqrt{\alpha_2}\boldsymbol{W}_2^T]$, and $\widetilde{\boldsymbol{X}}^T = [\sqrt{\alpha_1}\boldsymbol{X}_1^T \sqrt{\alpha_2}\boldsymbol{X}_2^T]$,

$$\min_{\widehat{\boldsymbol{H}}_1\geq 0} \left\|\begin{bmatrix}\sqrt{\beta_1}\boldsymbol{H}^T \\ \sqrt{\gamma_1}\boldsymbol{I}_k\end{bmatrix}\widehat{\boldsymbol{H}}_1 - \begin{bmatrix}\sqrt{\beta_1}\boldsymbol{S}_1 \\ \sqrt{\gamma_1}\boldsymbol{H}\end{bmatrix}\right\|_F^2, \tag{16}$$

$$\min_{\boldsymbol{W}^T\geq 0} \left\|\mathbb{M}^T \circ (\boldsymbol{H}^T\boldsymbol{W}^T - \boldsymbol{P}^T)\right\|_F^2, \tag{17}$$

$$\min_{\boldsymbol{H}\geq 0} \left\|\begin{bmatrix}\widetilde{\boldsymbol{W}}\boldsymbol{H} \\ \sqrt{\beta_1}\widehat{\boldsymbol{H}}_1{}^T\boldsymbol{H} \\ \sqrt{\gamma_1}\boldsymbol{H} \\ \mathbb{M} \circ (\boldsymbol{W}\boldsymbol{H})\end{bmatrix} - \begin{bmatrix}\widetilde{\boldsymbol{X}} \\ \sqrt{\beta_1}\boldsymbol{S}_1 \\ \sqrt{\gamma_1}\widehat{\boldsymbol{H}}_1 \\ \mathbb{M} \circ \boldsymbol{P}\end{bmatrix}\right\|_F^2 . \tag{18}$$

We use the block principal pivoting (BPP) method [15] to efficiently solve (15) and (16). On the other hand, we update $\boldsymbol{W}^T$ and $\boldsymbol{H}$ by columns to correctly incorporate the effect of the masking matrix $\mathbb{M}$ in (17) and (18). To solve (17), we update each column j of $\boldsymbol{W}^T$ by

$$\boldsymbol{W}^T(:,j) \leftarrow$$
$$\operatorname*{argmin}_{\boldsymbol{W}^T(:,j)\geq 0} \left\|D(\mathbb{M}^T(:,j))\boldsymbol{H}^T\boldsymbol{W}^T(:,j) - D(\mathbb{M}^T(:,j))\boldsymbol{P}^T(:,j)\right\|_F^2 \tag{19}$$

where $D(\mathbf{z})$ denotes the diagonal matrix whose diagonal entries are defined by the given vector $\mathbf{z}$, i.e., $D(\mathbf{z}) = [d_{ij}]_{n\times n}$ and $d_{ii} = z_i$ where $\mathbf{z} \in \mathbb{R}^n$ and z_i is the i-th component of $\mathbf{z}$. Also, to solve (18), we update each column of $\boldsymbol{H}$ by

$$\boldsymbol{H}(:,j) \leftarrow$$
$$\operatorname*{argmin}_{\boldsymbol{H}(:,j)\geq 0} \left\|\begin{bmatrix}\widetilde{\boldsymbol{W}} \\ \sqrt{\beta_1}\widehat{\boldsymbol{H}}_1{}^T \\ \sqrt{\gamma_1}\boldsymbol{I}_k \\ D(\mathbb{M}(:,j))\boldsymbol{W}\end{bmatrix}\boldsymbol{H}(:,j) - \begin{bmatrix}\widetilde{\boldsymbol{X}}(:,j) \\ \sqrt{\beta_1}\boldsymbol{S}_1(:,j) \\ \sqrt{\gamma_1}\widehat{\boldsymbol{H}}_1(:,j) \\ D(\mathbb{M}(:,j))\boldsymbol{P}(:,j)\end{bmatrix}\right\|_F^2 . \tag{20}$$

To speed up the update of $\boldsymbol{W}^T$ and $\boldsymbol{H}$, we group the columns of $\boldsymbol{W}^T$ and $\boldsymbol{H}$ so that we can apply the BPP method [15] per group instead of updating each column of $\boldsymbol{W}^T$ and $\boldsymbol{H}$ one by one. Let us represent (20) as follows:

$$\boldsymbol{H}(:,j) \leftarrow \operatorname*{argmin}_{\boldsymbol{H}(:,j)\geq 0} \|\boldsymbol{C}_j\boldsymbol{H}(:,j) - \mathbf{q}_j\|_F^2 , \tag{21}$$

where $\boldsymbol{C}_j$ depends on the column j of the matrix $\mathbb{M}$. Let $\mathcal{J}$ indicate a set of indices of the columns of $\boldsymbol{H}$ that have the same $\boldsymbol{C}_j$ which we will denote as $\boldsymbol{C}$. Also, let $\boldsymbol{Q} = [\mathbf{q}_1, \cdots, \mathbf{q}_n]$. Then, we can apply

$$\boldsymbol{H}(:,\mathcal{J}) \leftarrow \operatorname*{argmin}_{\boldsymbol{H}(:,\mathcal{J})\geq 0} \|\boldsymbol{C}\boldsymbol{H}(:,\mathcal{J}) - \boldsymbol{Q}(:,\mathcal{J})\|_F^2 . \tag{22}$$

Similarly, we update $\boldsymbol{W}^T$ by using this grouping strategy. When we incorporate partial supervision using $\boldsymbol{P}$, the masking matrix $\mathbb{M}$ has only few distinct columns and rows, which allows our grouping strategy to significantly accelerate the computations.

We can solve (13) by appropriately handling the masking matrix $\mathbb{Z}$ in a similar way. Assume that we have $\boldsymbol{X}_1$, $\boldsymbol{X}_2$, and $\boldsymbol{S}_1$. Then, (13) becomes

$$\min_{(\widetilde{\boldsymbol{W}},\boldsymbol{H},\widehat{\boldsymbol{H}}_1,\overline{\boldsymbol{H}})\geq 0} \left\|\widetilde{\boldsymbol{X}} - \widetilde{\boldsymbol{W}}\boldsymbol{H}\right\|_F^2 + \beta_1 \left\|\boldsymbol{S}_1 - \widehat{\boldsymbol{H}}_1{}^T\boldsymbol{H}\right\|_F^2 +$$
$$\gamma_1 \left\|\widehat{\boldsymbol{H}}_1 - \boldsymbol{H}\right\|_F^2 + \left\|\mathbb{Z} \circ (\boldsymbol{S} - \overline{\boldsymbol{H}}^T\boldsymbol{H})\right\|_F^2 + \mu \left\|\overline{\boldsymbol{H}} - \boldsymbol{H}\right\|_F^2 \tag{23}$$

where $\widetilde{\boldsymbol{X}} = [\sqrt{\alpha_1}\boldsymbol{X}_1; \sqrt{\alpha_2}\boldsymbol{X}_2]$, and $\widetilde{\boldsymbol{W}} = [\sqrt{\alpha_1}\boldsymbol{W}_1; \sqrt{\alpha_2}\boldsymbol{W}_2]$. Notice that $\mathbb{Z}$ is a symmetric matrix by definition as discussed in Section 4.3. We can optimize (23) using a 4-block coordinate descent where each sub-problem is defined as:

$$\min_{\widetilde{\boldsymbol{W}}\geq 0} \left\|\boldsymbol{H}^T\widetilde{\boldsymbol{W}}^T - \widetilde{\boldsymbol{X}}^T\right\|_F^2, \tag{24}$$

$$\min_{\widehat{\boldsymbol{H}}_1\geq 0} \left\|\begin{bmatrix}\sqrt{\beta_1}\boldsymbol{H}^T \\ \sqrt{\gamma_1}\boldsymbol{I}_k\end{bmatrix}\widehat{\boldsymbol{H}}_1 - \begin{bmatrix}\sqrt{\beta_1}\boldsymbol{S}_1 \\ \sqrt{\gamma_1}\boldsymbol{H}\end{bmatrix}\right\|_F^2, \tag{25}$$

$$\min_{\overline{\boldsymbol{H}}\geq 0} \left\|\begin{bmatrix}\mathbb{Z} \circ (\boldsymbol{H}^T\overline{\boldsymbol{H}}) \\ \sqrt{\mu}\overline{\boldsymbol{H}}\end{bmatrix} - \begin{bmatrix}\mathbb{Z} \circ \boldsymbol{S} \\ \sqrt{\mu}\boldsymbol{H}\end{bmatrix}\right\|_F^2, \tag{26}$$

$$\min_{\boldsymbol{H}\geq 0} \left\|\begin{bmatrix}\widetilde{\boldsymbol{W}}\boldsymbol{H} \\ \sqrt{\beta_1}\widehat{\boldsymbol{H}}_1^T\boldsymbol{H} \\ \sqrt{\gamma_1}\boldsymbol{H} \\ \mathbb{Z} \circ (\overline{\boldsymbol{H}}^T\boldsymbol{H}) \\ \sqrt{\mu}\boldsymbol{H}\end{bmatrix} - \begin{bmatrix}\widetilde{\boldsymbol{X}} \\ \sqrt{\beta_1}\boldsymbol{S}_1 \\ \sqrt{\gamma_1}\widehat{\boldsymbol{H}}_1 \\ \mathbb{Z} \circ \boldsymbol{S} \\ \sqrt{\mu}\overline{\boldsymbol{H}}\end{bmatrix}\right\|_F^2 . \tag{27}$$

While we can directly use the block principal pivoting algorithm [15] to solve (24) and (25), we update $\overline{\boldsymbol{H}}$ and $\boldsymbol{H}$ by columns to correctly consider the effect of the masking matrix $\mathbb{Z}$. To solve (26), we update each column of $\overline{\boldsymbol{H}}$ by

$$\overline{\boldsymbol{H}}(:,j) \leftarrow \underset{\overline{\boldsymbol{H}}(:,j)\geq 0}{\operatorname{argmin}} \left\| \begin{bmatrix} D(\mathbb{Z}(:,j))\boldsymbol{H}^T \\ \sqrt{\mu}\boldsymbol{I}_k \end{bmatrix} \overline{\boldsymbol{H}}(:,j) - \begin{bmatrix} D(\mathbb{Z}(:,j))\boldsymbol{S}(:,j) \\ \sqrt{\mu}\boldsymbol{H}(:,j) \end{bmatrix} \right\|_F^2 \tag{28}$$

Also, to solve (27), we update each column of $\boldsymbol{H}$ by

$$\boldsymbol{H}(:,j) \leftarrow \underset{\boldsymbol{H}(:,j)\geq 0}{\operatorname{argmin}} \left\| \begin{bmatrix} \widetilde{\boldsymbol{W}} \\ \sqrt{\beta_1}\widehat{\boldsymbol{H}}_1^T \\ \sqrt{\gamma_1}\boldsymbol{I}_k \\ D(\mathbb{Z}(:,j))\overline{\boldsymbol{H}}^T \\ \sqrt{\mu}\boldsymbol{I}_k \end{bmatrix} \boldsymbol{H}(:,j) - \begin{bmatrix} \widetilde{\boldsymbol{X}}(:,j) \\ \sqrt{\beta_1}\boldsymbol{S}_1(:,j) \\ \sqrt{\gamma_1}\widehat{\boldsymbol{H}}_1(:,j) \\ D(\mathbb{Z}(:,j))\boldsymbol{S}(:,j) \\ \sqrt{\mu}\overline{\boldsymbol{H}}(:,j) \end{bmatrix} \right\|_F^2 . \tag{29}$$

The above algorithm (`MEGA`) converges to a stationary point since `MEGA` follows the BCD framework and satisfies the conditions in the theorem by Bertsekas for the convergence analysis of the BCD [2,8,14]. Specifically, our objective function is continuously differentiable, the domains (nonnegative orthants) of each "block" ($\widetilde{\boldsymbol{W}}$, $\widehat{\boldsymbol{H}}_j$, $\boldsymbol{W}$, $\boldsymbol{H}$, $\overline{\boldsymbol{H}}$) are closed and convex (although no upper bound is imposed, we can assign a very large upper bound on each block so that the solution is the same without the explicit upper bound), and the minimum of each sub-problem, i.e., (15)–(18) and (24)–(27), is uniquely attained. Therefore, every limit point of the algorithm is a stationary point.

4.5 Time Complexity and Scalability Analysis

The time complexity of `MEGA` is dominated by solving the nonnegative least squares (NNLS) sub-problems (15)–(18) and (24)–(27). For an NNLS problem in the form

$$\min_{\boldsymbol{T}\geq 0} \|\boldsymbol{L}\boldsymbol{T} - \boldsymbol{R}\|_F^2, \tag{30}$$

where $\boldsymbol{L} \in \mathbb{R}_+^{n\times k}$, $\boldsymbol{T} \in \mathbb{R}_+^{k\times l'}$, and $\boldsymbol{R} \in \mathbb{R}_+^{n\times l'}$, the BPP algorithm we use has a time complexity upper bounded by $4kN + 2(n+l')k^2 + r[k^3/3 + 2(n+l')k^2]$ flops where N is the number of nonzeros in $\boldsymbol{R}$, and r is the number of iterations for searching the optimal active set (see [15] for more details). To analyze the scalability of `MEGA`, we measure the run time of `MEGA` on synthetic datasets with varying sizes. We assume that we have a hypergraph $\boldsymbol{S}_1$ with n objects and m hyperedges as well as $\boldsymbol{X}_1 \in \mathbb{R}_+^{l_1\times n}$ where l_1 is the number of features of the n objects. We provide partial supervision with $\boldsymbol{P}$ (discussed in Section 4.3) where 30% of the object labels are assumed to be observed. To create the ground-truth clusters, we divide the n nodes into k clusters where each cluster contains $n/k(1.5-\delta)$ nodes where δ is a uniformly random number between 0 and 1. To create $\boldsymbol{S}_1$, m randomly sized hyperedges are added such that the degree of a hyperedge varies from 2 to 10 and there are 60% within-cluster hyperedges to guarantee a meaningful clustering structure. We create $\boldsymbol{X}_1$ by a multivariate Gaussian distribution where each of the l_1 features is assumed to be independent and identically distributed. In Table 2, we show the run time in seconds and F1 (%) of `MEGA` on eight different synthetic datasets with varying k, n, m, and l_1 where `MEGA` is initialized by `hGraclus` (which will be discussed in Section 4.6), and thus, the run time also includes the run time of `hGraclus` (note that the run time of `hGraclus` is almost negligible since `hGraclus` is much faster than `MEGA`).

Table 2: Run time (in seconds) and F1 (%) of `MEGA` according to the number of clusters k, the number of nodes n, the number of hyperedges m, and the number of features l_1.

	k	n	m	l_1	Run Time	F1 (%)
SYN1	4	10,000	10,000	2	4.8	98.1
SYN2	4	50,000	500,000	10	78.3	100.0
SYN3	4	100,000	1,000,000	10	180.6	100.0
SYN4	4	500,000	1,000,000	1,000	1681.1	100.0
SYN5	8	100,000	200,000	10	90.5	100.0
SYN6	8	100,000	1,000,000	10	242.4	100.0
SYN7	8	500,000	1,000,000	10	2113.0	100.0
SYN8	8	500,000	1,000,000	1,000	2623.7	100.0

4.6 Initialization of `MEGA` using `hGraclus`

To apply the optimization algorithm (`MEGA`) presented in Section 4.4, we need to initialize $\boldsymbol{H}$. For example, we solve (15) given $\boldsymbol{H}$. A simple way to initialize $\boldsymbol{H}$ is to use a random real-valued matrix even though there can be a more sophisticated initialization [37]. We observe that initializing $\boldsymbol{H}$ in `MEGA` using `hGraclus` is beneficial for improving the clustering performance. As described in Section 3.3, `hGraclus` efficiently optimizes the hypergraph normalized cut by the multilevel weighted kernel K-Means. Indeed, `hGraclus` takes only 0.114 seconds to process a hypergraph with 42,889 nodes and 34,834 hyperedges. As described in Section 4.1, the hypergraph normalized cut objective can be reformulated as a SymNMF problem, and `MEGA` includes the SymNMF term to incorporate the hypergraph structure. Since both `hGraclus` and `MEGA` consider the hypergraph normalized cut, the solution that `hGraclus` produces can be considered to be a reasonable initialization of `MEGA`. The solution achieved by `hGraclus` is a $(0,1)$-matrix whereas $\boldsymbol{H}$ in `MEGA` is assumed to be a real-valued matrix. Thus, we add a small constant (we add 0.5 in our experiments) to the output of `hGraclus` to appropriately initialize $\boldsymbol{H}$.

Table 3 shows the clustering performance of `MEGA` with two different initializations: random and `hGraclus`. We use five synthetic datasets described in Table 2 as well as five real-world datasets described in Table 4. We present the average F1, accuracy, and NMI scores of 15 runs. Higher scores indicate better clustering results. Details about these measures are described in Section 6. We compute the gain of `hGraclus` initialization by dividing the difference between the score of `hGraclus` initialization and that of random initialization by the score of random initialization. We see that, overall, `MEGA` produces better clustering results with `hGraclus` initialization than random initialization.

5. RELATED WORK

Hypergraphs have been studied in various contexts (e.g., [4], [31], and [33]), where hypergraphs are used as important tools to encode higher-order relationships among objects. To appropriately handle hypergraphs, different types of graph Laplacian methods [29, 43] have been proposed to analyze the structure of a hypergraph. Among hypergraph clustering methods, `hMetis` [13] is a recognized method which is a multilevel partitioning method. While both `hMetis` and our `hGraclus` algorithm (presented in Section 3.3) are based on the multilevel optimization framework, they optimize different objectives. The goal of `hMetis` is to partition a hypergraph into k clusters of roughly equal sizes (since `hMetis` has been mainly studied in circuit partitioning) whereas `hGraclus` does not have explicit constraints on cluster sizes.

Table 3: The average F1, accuracy, and NMI scores of `MEGA` with two different initializations – random and `hGraclus`. The gain of `hGraclus` initialization is computed by (`hGraclus` - random)/random×100(%). We use the five synthetic datasets presented in Table 2 and the five real-world datasets presented in Table 4. Initializing `MEGA` with `hGraclus` leads to a better performance.

	F1 (↑)			Accuracy (↑)			NMI (↑)		
	random	`hGraclus`	Gain (%)	random	`hGraclus`	Gain (%)	random	`hGraclus`	Gain (%)
SYN1	87.19%	**98.06%**	12.47	89.84%	**97.83%**	8.89	85.17%	**93.88%**	10.23
SYN3	93.17%	**100.0%**	7.33	95.98%	**100.0%**	4.19	95.43%	**100.0%**	4.79
SYN5	94.84%	**100.0%**	5.44	96.93%	**100.0%**	3.17	97.50%	**100.0%**	2.56
SYN6	94.22%	**100.0%**	6.13	96.22%	**100.0%**	3.93	97.29%	**100.0%**	2.79
SYN7	78.19%	**100.0%**	27.89	87.14%	**100.0%**	14.76	92.69%	**100.0%**	7.89
CORA	65.48%	**68.58%**	4.73	68.32%	**71.29%**	4.35	45.04%	**46.98%**	4.31
DBLP5	84.41%	**86.89%**	2.94	83.70%	**85.77%**	2.47	63.50%	**65.96%**	3.87
GENE	57.16%	**58.50%**	2.34	60.29%	**61.22%**	1.54	20.55%	**21.36%**	3.94
DBLP10	**70.67%**	69.51%	-1.64	**74.52%**	73.87%	-0.87	60.77%	**61.46%**	1.14
QUERY	**57.97%**	57.55%	-0.72	55.70%	**55.80%**	0.18	**42.91%**	42.05%	-2.0
		Average Gain	6.69		Average Gain	4.26		Average Gain	3.95

For the traditional graph clustering, various models have been proposed to exploit not only graph structure but also available node attributes [30,39,44] even though these models are not designed to handle hypergraphs. Also, spectral ensemble clustering method [21] has been proposed to perform a robust ensemble clustering. We note that these clustering methods are unsupervised learning methods whereas `MEGA` can be used as either an unsupervised method (discussed in Section 4.2) or a semi-supervised method (discussed in Section 4.3).

Among a number of semi-supervised learning methods, a label propagation-based method [42] is well-known. Also, a maximum margin-based method [41] and a matrix completion-based method [40] have been proposed. Recently, a partition level constrained clustering method [22] also has been studied. We compare `MEGA` with these methods even though these semi-supervised learning methods do not take into account multi-view clustering.

There are very few methods that simultaneously consider semi-supervised learning and multi-view clustering. We note that the `SMVC` [10], `MLAN` [27], `SMACD` [9] methods are semi-supervised multi-view clustering methods but these methods are not specially designed for handling hypergraphs. In Section 6, we show that `MEGA` significantly outperforms `MLAN` and `SMACD` in terms of identifying the ground-truth clusters. We could not include `SMVC` as a baseline in our experiments because the code is not available.

Many different types of matrix factorization models [7] have been proposed to conduct consensus clustering [20], multi-view clustering [6,23] and semi-supervised learning [19]. In particular, [20] proposes an NMF-based model to perform consensus clustering. While [20] aggregates different clustering results using the connectivity matrix, `MEGA` directly reflects the diverse attributes and relations in the objective function. On the other hand, [23] proposes an unsupervised multi-view clustering method via joint nonnegative matrix factorization. Among these matrix factorization methods, we include [6] as a baseline in our experiments because it is closely related to our proposed method.

6. EXPERIMENTAL RESULTS

We show the performance of `MEGA` on five real-world datasets and compare it with 13 different state-of-the-art methods.

6.1 Datasets

We use five different real-world datasets: QUERY, GENE, CORA, DBLP5, and DBLP10 which are shown in Table 4.

Table 4: Real-world Datasets

	No. of nodes	No. of hyperedges	k	Views
QUERY	481	15,762	6	$\boldsymbol{X}_1$, $\boldsymbol{X}_2$, $\boldsymbol{S}_1$, $\boldsymbol{P}$
GENE	2,014	2,023	4	$\boldsymbol{X}_1$, $\boldsymbol{S}_1$, $\boldsymbol{S}_2$, $\boldsymbol{P}$
CORA	2,485	2,485	7	$\boldsymbol{X}_1$, $\boldsymbol{S}_1$, $\boldsymbol{P}$
DBLP5	19,756	21,492	5	$\boldsymbol{X}_1$, $\boldsymbol{X}_2$, $\boldsymbol{S}_1$, $\boldsymbol{P}$
DBLP10	42,889	34,834	10	$\boldsymbol{X}_1$, $\boldsymbol{X}_2$, $\boldsymbol{S}_1$, $\boldsymbol{P}$

The QUERY dataset is the one we introduced in Section 1. We have a hypergraph $\boldsymbol{S}_1$ where a node indicates a query and a hyperedge indicates a query session (there are 481 queries and 15,762 query sessions), the embedding matrix $\boldsymbol{X}_1 \in \mathbb{R}_{+}^{128\times 481}$, and the clicked documents information $\boldsymbol{X}_2 \in \mathbb{R}_{+}^{298\times 481}$. We have six ground-truth clusters for the queries that are manually labeled by the engineers in NAVER who provided the dataset. The ground-truth clusters group the queries that should be handled together, e.g., the queries that are related to 'payment', or 'place' and so on.

For GENE [26], we have information about which genes are associated with which diseases. Since a gene can get involved in more than two diseases, the gene-disease associations can be modeled as a hypergraph where a node indicates a disease and a hyperedge indicates a gene. This hypergraph corresponds to $\boldsymbol{S}_1$ in our model (there are 2,014 diseases and 2,023 genes). Also, there is a tf-idf representation of the diseases which can be used as $\boldsymbol{X}_1 \in \mathbb{R}_{+}^{16{,}592\times 2{,}014}$ as well as the pairwise similarities between the diseases which is $\boldsymbol{S}_2 \in \mathbb{R}^{2{,}014\times 2{,}014}$. The ground-truth clusters of the diseases are defined in the OMIM database (`https://www.omim.org/`) where each ground-truth cluster corresponds to a specific type of phenotypes in biology.

For CORA [25], we construct a hypergraph based on the citation information where a node represents a paper and a hyperedge groups the given paper and the papers cited by the given paper (there are 2,485 papers). Since a paper can be well described by its list of references, we believed that this hypergraph representation $\boldsymbol{S}_1$ is useful to cluster the papers, which turns out to be true in our experiments. Also, each paper is associated with a set of predefined keywords. This information is used for $\boldsymbol{X}_1 \in \mathbb{R}^{1{,}433\times 2{,}485}$ in our model. There are seven ground-truth clusters of the papers based on their topics.

We make DBLP5 and DBLP10 from [32]. We extract papers that belong to the top 5 (top 10) largest venues, and these venues are considered to be the ground-truth clusters. Based on the title and the abstract of each paper, we construct a tf-idf representation of the papers which corresponds to

$\boldsymbol{X}_1 \in \mathbb{R}_+^{7,116\times 21,492}$ for DBLP5 and $\boldsymbol{X}_1 \in \mathbb{R}_+^{11,245\times 34,834}$ for DBLP10. Using the citation information, we construct $\boldsymbol{X}_2 \in \{0,1\}^{21,492\times 21,492}$ for DBLP5 and $\boldsymbol{X}_2 \in \{0,1\}^{34,834\times 34,834}$ for DBLP10. Also, we construct a hypergraph $\boldsymbol{S}_1$ (19,756 nodes and 21,492 hyperedges for DBLP5 and 42,889 nodes and 34,834 hyperedges for DBLP10) based on the collaboration network where a node indicates an individual and a hyperedge indicates a paper, i.e., the co-authors of a paper are associated with the same hyperedge. In this setting, our problem is to cluster the hyperedges instead of the nodes because a paper corresponds to a hyperedge. To solve this hyperedge clustering problem, we compute $\widetilde{\boldsymbol{A}}$ as discussed in Section 3.2. On DBLP5, we empirically observe that $\boldsymbol{S}_1$, $\boldsymbol{X}_1$, and $\boldsymbol{P}$ are critical factors that affect the clustering quality whereas $\boldsymbol{X}_2$ is a redundant feature set.

6.2 Baseline Methods and Experimental Setup

We compare the performance of MEGA with 13 different state-of-the-art methods listed below. We add a brief note to each method to summarize the characteristics of the method.

- hGraclus is our multilevel hypergraph clustering discussed in Section 3.3. **[hypergraph structure only]**
- hMetis [13] is a hypergraph clustering method that partitions a hypergraph into k balanced clusters. **[hypergraph structure only]**
- SPC [43] is a spectral hypergraph partitioning method. **[hypergraph structure only]**
- SWS [28] is a hypergraph clustering method that allows large hyperedges. **[hypergraph structure only]**
- PCLDC [39] is a discriminative model which combines the link and content analysis for clustering. **[multi-view clustering]**
- JNMF [6] is a joint NMF-based clustering which incorporates both graph structure and attributes. **[multi-view clustering]**
- SEC [21] is a spectral ensemble clustering method. **[multi-view clustering]**
- CMMC [41] is a constrained maximum margin clustering method. **[semi-supervised learning with $\boldsymbol{S}$]**
- MCCC [40] is a semi-supervised clustering with pairwise similarity matrix completion. **[semi-supervised learning with $\boldsymbol{S}$]**
- LGC [42] is a label propagation-based semi-supervised model. **[semi-supervised learning with $\boldsymbol{P}$]**
- PLCC [22] is a partition level constrained clustering method. **[semi-supervised learning with $\boldsymbol{P}$]**
- SMACD [9] is a tensor factorization-based method for semi-supervised multi-aspect community detection. **[multi-view & semi-supervised learning with $\boldsymbol{P}$]**
- MLAN [27] is a multi-view and semi-supervised model with local structure learning. **[multi-view & semi-supervised learning with $\boldsymbol{P}$]**

We use publicly available software for the baseline methods or get the software from the authors. We use the default parameters the software provides because we also use the default parameters in MEGA. For hMetis, we use shmetis, and set $UBfactor = 1$. For SWS [28], we set the number of iterations to be 100 for QUERY, GENE, and CORA, and to be 10 for DBLP5 and DBLP10 since SWS takes more than 5 hours if the number of iterations is set to be 100 on these two large datasets. For MCCC, we set $C = n/\sqrt{k}$ by reading Theorem 2 in [40]. For LGC, we set $\alpha = 0.99$ as suggested in [42]. For MLAN, we normalize the input matrices to balance the effect of each view. Among the baseline methods, only hGraclus, hMetis, SPC, and SWS assume that the input is a hypergraph. To process a hypergraph using the other baseline methods, we compute $\widehat{\boldsymbol{A}}$ or $\widetilde{\boldsymbol{A}}$ which are described in Section 3.2. Since SMACD and MLAN are multi-view semi-supervised methods, these methods take all the available features and partially observed labels like MEGA.

There are two different types of semi-supervised learning methods: (i) ones that assume the labels of the objects are directly observed, and (ii) ones that assume only a set of pairwise constraints is given instead of the direct labels. In MEGA, we can handle both of these cases; we handle (i) by introducing $\boldsymbol{P}$ whereas we handle (ii) by introducing $\boldsymbol{S}$ as described in Section 4.3. In our experiments, we use $\boldsymbol{P}$ to incorporate semi-supervision in MEGA.

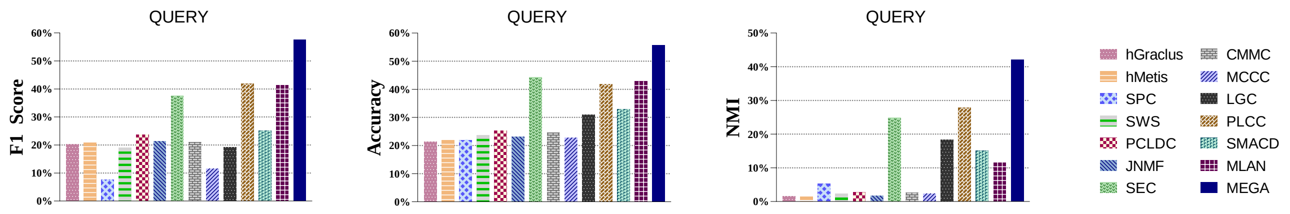
For the semi-supervised methods, we use five different masks, i.e., we randomly generate the masking matrices $\mathbb{M}$ or $\mathbb{Z}$ (described in Section 4.3) for five times. We run each method three times for each mask (i.e., 15 runs in total). For the unsupervised methods, we run each method five times. We do not report a result if a method takes more than 10 hours to complete five runs. When $\boldsymbol{P}$ is used for a semi-supervised method, we assume that 30% of the object labels are observed. For the models that require $\boldsymbol{S}$ (CMMC and MCCC), we provide 500 pairwise constraints for QUERY, and 5,000 pairwise constraints for GENE and CORA since these methods usually require $\mathcal{O}(n)$ constraints. Both of CMMC and MCCC took more than 10 hours on DBLP5 and DBLP10.

For MEGA, we set $\alpha_i = 1$ for all $i = 1, \cdots p$, $\beta_j = 1$ for all $j = 1, \cdots, q$, and $\gamma_j = \max(\boldsymbol{S}_j)$ as described in Section 4.3. In MEGA, we initialize $\boldsymbol{H}$ using hGraclus as described in Section 4.6. Among the baseline methods, PCLDC and JNMF allow us to provide an initial $\boldsymbol{H}$. We provide the same $\boldsymbol{H}$ we used for initializing MEGA with these methods. Since all the methods require the number of clusters as an input, we provide the ground-truth number of clusters with each of the methods. To measure the clustering performance, we compute the F1, accuracy, and the normalized mutual information (NMI) scores which are also used in [27] and [35]. We compute the best matching between the ground-truth clusters and the algorithmic clusters to compute these metrics.

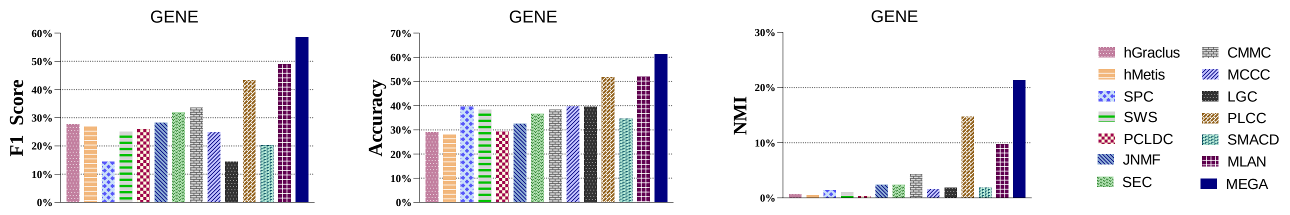
6.3 Quality Evaluation

Figure 2 shows the average F1, accuracy, and NMI scores of the 14 methods (MEGA and the 13 baseline methods introduced in Section 6.2) on the five real-world datasets introduced in Section 6.1. On CORA, MCCC returned an SVD error message. On DBLP5 and DBLP10, PCLDC, CMMC, MCCC, and SMACD took more than 10 hours to complete five runs. On DBLP10, LGC and MLAN required more than 128GB memory, so we could not report the results of these methods.

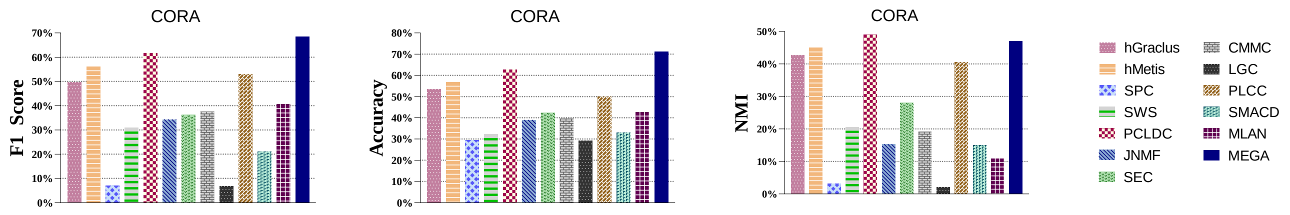
Let us first briefly mention the performances of the hypergraph clustering methods that only consider the hypergraph structure: hGraclus, hMetis, SPC, and SWS. While we focus on the hypergraph normalized cut and run time in Table 1 to evaluate these methods, Figure 2 shows the similarity between the algorithmic solutions and the ground-truth clusters. Even though the hypergraph normalized cut is one of the standard objectives for hypergraph clustering, and generally, an objectively better solution is expected to be closer to the ground-truth, it is not theoretically guaranteed that an objectively better solution always yields a closer solution to the ground-truth. This is why there can be some discrepancies between Table 1 and Figure 2.



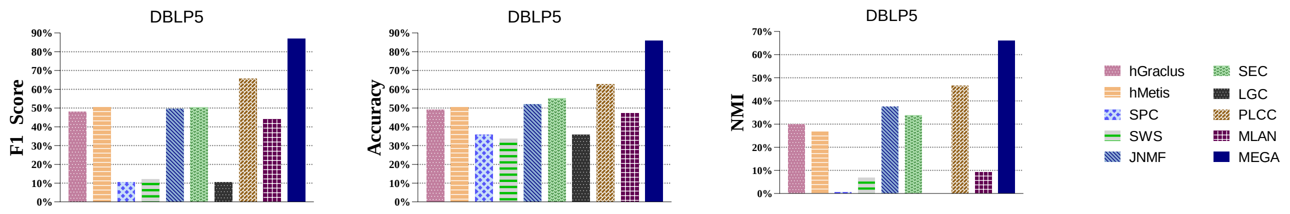
(a) F1, Accuracy, and NMI on QUERY



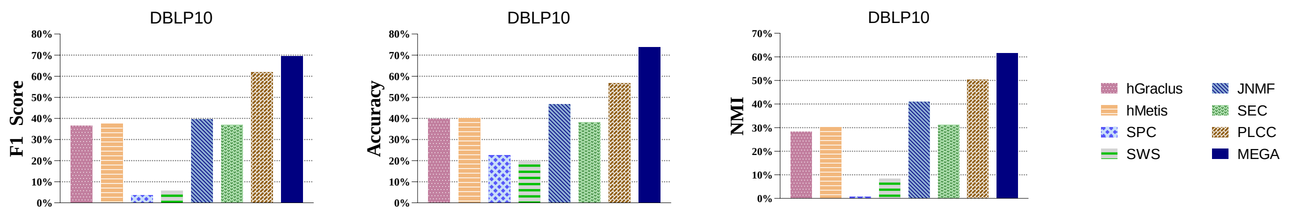
(b) F1, Accuracy, and NMI on GENE



(c) F1, Accuracy, and NMI on CORA



(d) F1, Accuracy, and NMI on DBLP5



(e) F1, Accuracy, and NMI on DBLP10

Figure 2: Higher F1, accuracy, and NMI scores indicate better clustering results. In terms of identifying the ground-truth clusters, `MEGA` outperforms the other 13 different state-of-the-art methods on the five real-world datasets presented in Section 6.1. On CORA, `MCCC` returned an SVD error message. On DBLP5 and DBLP10, `PCLDC`, `CMMC`, `MCCC`, and `SMACD` took more than 10 hours to complete five runs. On DBLP10, `LGC` and `MLAN` required more than 128GB memory, so we could not report the results of these methods.

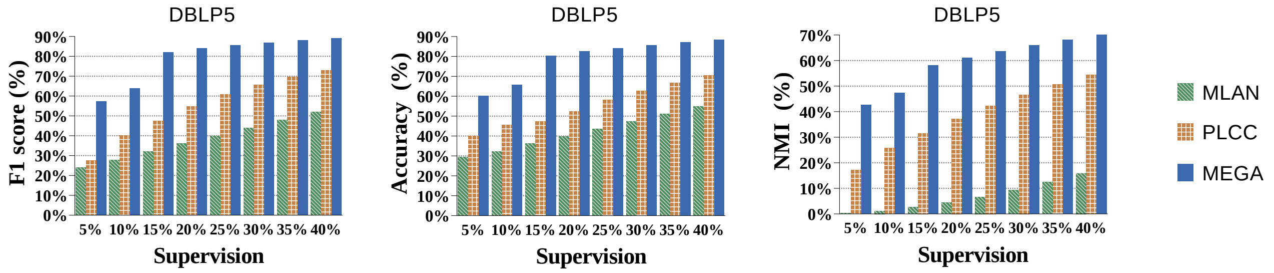


Figure 3: The average F1, accuracy, and NMI scores of MLAN, PLCC, and MEGA with different levels of supervision on DBLP5. MEGA achieves better clustering performance than the other semi-supervised methods at all different levels of supervision.

Table 5: Clustering performance of CMMC and MEGA on CORA with different numbers of pairwise constraints.

	F1 (%)		Accuracy (%)		NMI (%)	
Constraints	CMMC	MEGA	CMMC	MEGA	CMMC	MEGA
3,000	33.78	**60.61**	35.61	**62.77**	15.71	**43.58**
4,000	38.43	**62.45**	40.32	**64.65**	19.91	**45.65**
5,000	37.58	**63.41**	39.48	**65.75**	19.06	**48.04**
6,000	39.08	**65.40**	41.45	**67.95**	20.26	**50.64**
7,000	35.96	**66.98**	38.44	**69.56**	19.28	**52.57**

Table 6: Performance of MEGA with $\boldsymbol{P}$ and $\boldsymbol{S}$ on CORA. We use $\boldsymbol{P}$ with 30% supervision, and use $\boldsymbol{S}$ with 7,000 pairwise constraints.

	F1 (%)	Accuracy (%)	NMI (%)
MEGA-P	68.58 ± 0.97	71.29 ± 1.06	46.98 ± 1.20
MEGA-S	66.98 ± 1.29	69.56 ± 1.18	52.57 ± 1.19

In Figure 2, we see that MEGA outperforms the other 13 different state-of-the-art methods. It is encouraging that MEGA shows good performance even though all the parameters are set to be ones (i.e., we do not tune the parameters of MEGA). By observing that MEGA shows the best performance in identifying the ground-truth clusters, we note that MEGA is able to appropriately take into account multiple views of the data and properly incorporate partial supervision into the clustering model. Also, the MEGA objective function proposed in (12) is able to more effectively capture the structure of the ground-truth clusters than the other baselines.

We test the performance of the semi-supervised methods by varying the levels of supervision provided by $\boldsymbol{P}$ and $\boldsymbol{S}$. Figure 3 shows the average F1, accuracy, and NMI scores of MLAN, PLCC, and MEGA which are the models that incorporate the supervision using $\boldsymbol{P}$ on DBLP5. We vary the supervision from 5% to 40%. As the level of supervision increases, the performance of these methods increases as expected. We see that MEGA achieves better clustering results than MLAN and PLCC for all different levels of supervision. We conduct similar analysis for the semi-supervised methods that require pairwise constraints (i.e., the semi-supervised methods with $\boldsymbol{S}$). For MEGA, we use $\boldsymbol{S}$ for this analysis, and set all the parameters to be ones. Recall that MEGA can incorporate partial supervision using either $\boldsymbol{P}$ or $\boldsymbol{S}$ as described in Section 4.3. Table 5 shows the average F1, accuracy, and NMI scores on CORA. Since MCCC returned an error on CORA, we focus on CMMC. We note that MEGA outperforms CMMC given the same number of pairwise constraints. On CORA, when we use $\boldsymbol{S}$ with 7,000 pairwise constraints for MEGA, denoted by MEGA-S in Table 6, its performance is close to that of MEGA-P where we use $\boldsymbol{P}$ with 30% supervision shown in Table 6.

We set all the parameters of MEGA to be ones in the above experiments. Now, we test how the performance of MEGA varies according to the parameters. Table 7 shows the performance of MEGA with different parameters and the performance of the two most competitive baseline methods, PLCC and JNMF, on DBLP5. For MEGA, we vary the three parameters, α_1, α_2, and β_1 while all other parameters are set to be ones. Table 8 shows the performance of MEGA, and the two most competitive baseline methods, MLAN and PLCC, on GENE. For MEGA, we vary the parameters, α_1, β_1, and β_2, and all other parameters are set to be ones. We see that the performance of MEGA does not largely fluctuate depending on the parameters, and MEGA consistently outperforms the baseline methods.

Table 7: Performance of MEGA with different parameters and the two most competitive baseline methods on DBLP5

	MEGA						PLCC	JNMF
X_1 (α_1)	0.3	0.3	0.3	1.0	1.0	1.0		
X_2 (α_2)	0.3	0.3	1.0	0.3	1.0	1.0		
S_1 (β_1)	0.3	1.0	1.0	1.0	0.3	1.0		
F1 (%)	87.48	89.05	89.12	86.87	86.42	86.89	65.69	49.61
ACC (%)	86.45	87.98	88.05	85.75	85.30	85.77	62.71	51.98
NMI (%)	66.88	69.97	70.11	65.92	65.05	65.96	46.62	37.64

Table 8: Performance of MEGA with different parameters and the two most competitive baseline methods on GENE

	MEGA						MLAN	PLCC
X_1 (α_1)	0.3	0.3	0.3	1.0	1.0	1.0		
S_1 (β_1)	0.3	0.3	1.0	0.3	1.0	1.0		
S_2 (β_2)	0.3	1.0	1.0	1.0	0.3	1.0		
F1 (%)	57.65	56.95	56.32	56.33	57.27	58.50	48.93	43.33
ACC (%)	60.10	59.47	58.41	59.92	60.41	61.22	51.94	51.77
NMI (%)	23.70	22.07	18.90	19.76	20.41	21.36	9.72	14.74

Table 9: Run time (in seconds) on QUERY.

hGraclus	0.005	SPC	0.131	SEC	0.160	LGC	0.168
MEGA	0.544	JNMF	0.626	MLAN	1.06	hMetis	1.23
PLCC	3.92	PCLDC	6.50	MCCC	46.9	SWS	51.3
SMACD	727.5	CMMC	1279.3				

Finally, Table 9 shows the run time (in seconds) of MEGA and the 13 baseline methods on QUERY. We see that hGraclus is the fastest method whereas MEGA is the fifth fastest method among the 14 methods.

7. CONCLUSIONS AND FUTURE WORK

We develop a semi-supervised clustering method for hypergraphs in which we can easily incorporate multiple attributes and content of the objects as well as diverse relationships among the objects. According to our extensive tests, the proposed method MEGA significantly outperforms all other state-of-the-art methods on real-world datasets in terms of identifying the ground-truth clusters. We plan to extend our model to non-exhaustive, overlapping clustering [34, 35] and develop more scalable solution procedures.

Acknowledgment

This work was supported by NAVER Corp. and NRF of Korea (2019R1C1C1008956, 2018R1A5A1059921) to J. Whang, and by the US NSF grant OAC-1642410 to H. Park.

8. REFERENCES

[1] S. Basu, M. Bilenko, and R. Mooney. A probabilistic framework for semi-supervised clustering. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 59–68, 2004.

[2] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

[3] S. Bickel and T. Scheffer. Multi-view clustering. In *Proceedings of the 4th IEEE International Conference on Data Mining*, pages 19–26, 2004.

[4] L. Chen, Y. Gao, Y. Zhang, S. Wang, and B. Zheng. Scalable hypergraph-based image retrieval and tagging system. In *Proceedings of the 34th IEEE International Conference on Data Engineering*, pages 257–268, 2018.

[5] I. S. Dhillon, Y. Guan, and B. Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1944–1957, 2007.

[6] R. Du, B. Drake, and H. Park. Hybrid clustering based on content and connection structure using joint nonnegative matrix factorization. *Journal of Global Optimization*, pages 1–17, 2017.

[7] R. Du, D. Kuang, B. Drake, and H. Park. DC-NMF: Nonnegative matrix factorization based on divide-and-conquer for fast clustering and topic modeling. *Journal of Global Optimization*, 68(4):777–798, 2017.

[8] L. Grippo and M. Sciandrone. On the convergence of the block nonlinear gauss–seidel method under convex constraints. *Operations Research Letters*, pages 127–136, 2000.

[9] E. Gujral and E. E. Papalexakis. SMACD: semi-supervised multi-aspect community detection. In *Proceedings of the 2018 SIAM International Conference on Data Mining*, pages 702–710, 2018.

[10] S. Günnemann, I. Färber, M. Rüdiger, and T. Seidl. SMVC: Semi-supervised multi-view clustering in subspace projections. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 253–262, 2014.

[11] K. Han, F. Gui, X. Xiao, J. Tang, Y. He, Z. Cao, and H. Huang. Efficient and effective algorithms for clustering uncertain graphs. *PVLDB*, 12(6):667–680, 2019.

[12] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, pages 359–392, 1999.

[13] G. Karypis and V. Kumar. Multilevel k-way hypergraph partitioning. In *Proceedings of the 36th Annual ACM/IEEE Design Automation Conference*, pages 343–348, 1999.

[14] J. Kim, Y. He, and H. Park. Algorithms for nonnegative matrix and tensor factorizations: A unified view based on block coordinate descent framework. *Journal of Global Optimization*, pages 285–319, 2014.

[15] J. Kim and H. Park. Fast nonnegative matrix factorization: An active-set-like method and comparisons. *SIAM Journal on Scientific Computing*, pages 3261–3281, 2011.

[16] D. Kuang, C. Ding, and H. Park. Symmetric nonnegative matrix factorization for graph clustering. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, pages 106–117, 2012.

[17] D. Kuang, S. Yun, and H. Park. SymNMF: Nonnegative low-rank approximation of a similarity matrix for graph clustering. *Journal of Global Optimization*, 62(3):545–574, 2015.

[18] B. Kulis, S. Basu, I. S. Dhillon, and R. Mooney. Semi-supervised graph clustering: a kernel approach. *Machine Learning*, pages 1–22, 2009.

[19] H. Lee, J. Yoo, and S. Choi. Semi-supervised nonnegative matrix factorization. *IEEE Signal Processing Letters*, pages 4–7, 2010.

[20] T. Li, C. Ding, and M. Jordan. Solving consensus and semi-supervised clustering problems using nonnegative matrix factorization. In *Proceedings of the 7th IEEE International Conference on Data Mining*, pages 577–582, 2007.

[21] H. Liu, T. Liu, J. Wu, D. Tao, and Y. Fu. Spectral ensemble clustering. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 715–724, 2015.

[22] H. Liu, Z. Tao, and Y. Fu. Partition level constrained clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 2469–2483, 2018.

[23] J. Liu, C. Wang, J. Gao, and J. Han. Multi-view clustering via joint nonnegative matrix factorization. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 252–260, 2013.

[24] U. Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, pages 395–416, 2007.

[25] J. Motl and O. Schulte. The CTU prague relational learning repository. `relational.fit.cvut.cz/dataset/CORA`, 2015.

[26] N. Natarajan and I. S. Dhillon. Inductive matrix completion for predicting gene–disease associations. *Bioinformatics*, pages i60–i68, 2014.

[27] F. Nie, G. Cai, and X. Li. Multi-view clustering and semi-supervised classification with adaptive neighbours. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, pages 2408–2414, 2017.

[28] P. Purkait, T. Chin, A. Sadri, and D. Suter. Clustering with hypergraphs: The case for large hyperedges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1697–1711, 2017.

[29] S. Saito, D. Mandic, and H. Suzuki. Hypergraph p-Laplacian: A differential geometry view. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 3984–3991, 2018.

[30] Y. Sun, C. C. Aggarwal, and J. Han. Relation strength-aware clustering of heterogeneous information networks with incomplete attributes. *PVLDB*, 5(5):394–405, 2012.

[31] S. Tan, Z. Guan, D. Cai, X. Qin, J. Bu, and C. Chen. Mapping users across networks by manifold alignment on hypergraph. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, pages 159–165, 2014.

[32] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su. ArnetMiner: Extraction and mining of academic social

networks. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 990–998, 2008.

[33] Z. Tian, T. Hwang, and R. Kuang. A hypergraph-based learning algorithm for classifying gene expression and arrayCGH data with prior knowledge. *Bioinformatics*, pages 2831–2838, 2009.

[34] J. J. Whang, D. F. Gleich, and I. S. Dhillon. Overlapping community detection using neighborhood-inflated seed expansion. *IEEE Transactions on Knowledge and Data Engineering*, 28(5):1272–1284, 2016.

[35] J. J. Whang, Y. Hou, D. F. Gleich, and I. S. Dhillon. Non-exhaustive, overlapping clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(11):2644–2659, 2019.

[36] J. J. Whang, X. Sui, and I. S. Dhillon. Scalable and memory-efficient clustering of large-scale social networks. In *Proceedings of the 12th IEEE International Conference on Data Mining*, pages 705–714, 2012.

[37] S. Wild, J. Curry, and A. Dougherty. Improving non-negative matrix factorizations through structured initialization. *Pattern Recognition*, pages 2217–2232, 2004.

[38] Z. Xu, Y. Ke, Y. Wang, H. Cheng, and J. Cheng. A model-based approach to attributed graph clustering. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 505–516, 2012.

[39] T. Yang, R. Jin, Y. Chi, and S. Zhu. Combining link and content for community detection: A discriminative approach. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 927–936, 2009.

[40] J. Yi, L. Zhang, R. Jin, Q. Qian, and A. Jain. Semi-supervised clustering by input pattern assisted pairwise similarity matrix completion. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1400–1408, 2013.

[41] H. Zeng and Y. Cheung. Semi-supervised maximum margin clustering with pairwise constraints. *IEEE Transactions on Knowledge and Data Engineering*, pages 926–939, 2011.

[42] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Proceedings of the 16th International Conference on Neural Information Processing Systems*, pages 321–328, 2003.

[43] D. Zhou, J. Huang, and B. Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. In *Proceedings of 19th Advances in Neural Information Processing Systems*, pages 1601–1608, 2006.

[44] Y. Zhou, H. Cheng, and J. X. Yu. Graph clustering based on structural/attribute similarities. *PVLDB*, 2(1):718–729, 2009.