Parameterized Correlation Clustering in Hypergraphs and Bipartite Graphs

Nate Veldt
Cornell University
Center for Applied Mathematics
nveldt@cornell.edu

Anthony Wirth
The University of Melbourne
Computing and Information Systems
awirth@unimelb.edu.au

David F. Gleich
Purdue University
Department of Computer Science
dgleich@purdue.edu

ABSTRACT

Motivated by applications in community detection and dense subgraph discovery, we consider new clustering objectives in hypergraphs and bipartite graphs. These objectives are parameterized by one or more *resolution parameters* in order to enable diverse knowledge discovery in complex data.

For both hypergraph and bipartite objectives, we identify relevant parameter regimes that are equivalent to existing objectives and share their (polynomial-time) approximation algorithms. We first show that our parameterized hypergraph correlation clustering objective is related to higher-order notions of normalized cut and modularity in hypergraphs. It is further amenable to approximation algorithms via hyperedge expansion techniques.

Our parameterized bipartite correlation clustering objective generalizes standard unweighted bipartite correlation clustering, as well as the bicluster deletion problem. For a certain choice of parameters it is also related to our hypergraph objective. Although in general it is NP-hard, we highlight a parameter regime for the bipartite objective where the problem reduces to the bipartite matching problem and thus can be solved in polynomial time. For other parameter settings, we present several approximation algorithms using linear program rounding techniques. These results allow us to introduce the first constant-factor approximation for bicluster deletion, the task of removing a minimum number of edges to partition a bipartite graph into disjoint bi-cliques.

In several experimental results, we highlight the flexibility of our framework and the diversity of results that can be obtained in different parameter settings. This includes clustering bipartite graphs across a range of parameters, detecting motif-rich clusters in an email network and a food web, and forming clusters of retail products in a product review hypergraph, that are highly correlated with known product categories.

KEYWORDS

hypergraphs, bipartite graphs, correlation clustering

ACM Reference Format:
Nate Veldt, Anthony Wirth, and David F. Gleich. 2020. Parameterized Correlation Clustering in Hypergraphs and Bipartite Graphs. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3394486.3403238

KDD '20, August 23–27, 2020, Virtual Event, CA, USA

ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403238

1 INTRODUCTION

Finding sets of related objects in a large dataset, i.e., *clustering*, is one of the fundamental tasks in data mining and machine learning, and is often used as a first step in exploring and understanding a new dataset. When the data to be clustered is represented by a graph or network, the task is referred to as graph clustering or community detection [19, 45]. A good graph clustering is one in which nodes in the same cluster share many edges with each other, but nodes in different clusters share few edges. While these basic principles are shared by nearly all graph clustering techniques, there are many ways to formalize the notion of a graph cluster [19, 45, 54]. However, no one method or objective function is capable of solving all graph clustering tasks [41].

One outcome is that there are many graph clustering objectives that rely on one or more tunable *resolution* parameters, which can control the size, structure, or edge density of the clusters formed by optimizing the objective [7, 16, 43, 50, 54, 55]. In addition to providing a way to detect clusters at different resolutions in a graph, parametric clustering objectives often make it possible to interpolate between other existing and commonly studied graph clustering objectives. Recently, we showed [54] that a number of popular graph clustering objectives such as modularity [39], normalized cut [47], and cluster deletion [46] can be captured as special cases of a parametric variant of correlation clustering [9].

Nearly all existing techniques for parametric graph clustering focus on a simple graph setting, where all nodes are of the same type and are inter-related by pairwise connections, represented by edges. However, graph and complex network datasets often have additional structure, which can be exploited for the purpose of more in-depth data analysis. As an example, there has been a recent surge of interest in *higher-order* methods for clustering [4, 10, 35, 36, 51, 53, 58–60]. These determine the clustering of the data not only via its graph edges, but also based on its motifs (small, frequently appearing subgraphs), or indeed based on hyperedges in a hypergraph. Motifs and hyperedges admit encoding multiway relationships between sets of three or more nodes. This provides a more faithful way to represent complex systems characterized by interactions that are inherently multiway. For example, in co-authorship datasets, papers are frequently written by more than two authors. Applications of higher-order and hypergraph clustering include image segmentation and computer vision problems [1, 30], circuit design and VLSI layout [23, 29], and bioinformatics [38, 49].

Bipartite graphs model interactions between two different types of objects. These have a close relationship with hypergraphs in general, as witnessed in the example of co-authorship data. In a hypergraph, each author is a node and the set of authors in each paper is represented by a hyperedge. In a bipartite graph, one set of nodes represents authors, the other set papers: nodes i and j are adjacent whenever person i is an author of paper j. Which representation is best depends on the task; importantly, either of these is more informative than a simple network in which each edge indicates whether that pair of authors have ever co-authored.

Just as there are many objective functions for graph clustering, many different objectives for clustering hypergraphs and bipartite graphs have been developed, each of which strikes different balances in terms of size and structure of output clusters [2, 5, 8, 15, 20, 21, 27, 30, 34, 35, 37, 60]. The prevalence and variety of different methods indicates that hypergraphs and bipartite graphs can also exhibit clustering structure at different resolutions. However, these existing methods for clustering hypergraphs and bipartite graphs largely ignore parametric clustering objectives. Thus, in this paper we present a rigorous framework for parametric clustering in these settings. Our objectives are based on parameterized versions of correlation clustering and we show how, in certain parameter regimes, our objectives are related to a number of these previous objectives for bipartite and hypergraph clustering. Furthermore, our methods come with new approximation results. In summary,

(1) We present HyperLam, a parametric hypergraph clustering objective that we prove is related to hypergraph generalizations of the normalized cut and modularity objectives.
(2) We present a parametric bipartite correlation clustering objective (PBCC), which captures standard bipartite correlation clustering and bicluster deletion [5] as special cases. We also prove that in certain parameter regimes it is equivalent to a variant of our HyperLam objective.
(3) We prove that HyperLam admits an $O(\log n)$ approximation by combining certain *expansion* techniques with approximation algorithms for correlation clustering in graphs. We also consider faster heuristic approaches based on applying greedy agglomeration methods.
(4) While PBCC is NP-hard in general, we prove that in a certain parameter regime it is equivalent to bipartite matching and can thus be solved in polynomial time.
(5) Via linear programming relaxation techniques, we show a number of approximation algorithm that apply to different parameter settings of PBCC, including the first constant factor approximation for bicluster deletion, the problem of partitioning a bipartite graph into disjoint bicliques by removing a minimum number of edges.

As a brief overview of our paper, we begin with small technical preliminaries on correlation clustering, graph clustering, and hypergraph clustering. Then we state our two new objectives for parametric hypergraph and bipartite clustering in Sections 3 and 4, and prove their equivalence with existing objectives. We discuss algorithms and heuristics in Section 5 before showing how these algorithms work in a variety of scenarios (Section 7).

2 PRELIMINARIES

We begin with technical preliminaries on correlation clustering, graph clustering, and hypergraph clustering.

2.1 Correlation Clustering

A standard weighted instance of correlation clustering is given by a graph $G = (V, W^+, W^-)$, where each pair of nodes $(i, j) \in V \times V$ with $i \neq j$ is associated with positive and negative weights $w_{ij}^+ \in W^+$ and $w_{ij}^- \in W^-$. Given this input, the objective is to minimize the weight of *mistakes* or *disagreements*. If nodes i and j are clustered together, they incur a mistake with penalty w_{ij}^-, and if they are separated, they incur a mistake with penalty w_{ij}^+. For instances where at most one of (w_{ij}^+, w_{ij}^-) is non-zero, this can be viewed as a clustering problem in a signed graph. The objective can formally be stated as a binary linear program (BLP):

$$\begin{array}{lll} \text{minimize} & \sum_{i<j} w_{ij}^+ x_{ij} + w_{ij}^-(1 - x_{ij}) & \\ \text{subject to} & x_{ij} \leq x_{ik} + x_{jk} & \text{for all } i, j, k \\ & x_{ij} \in \{0, 1\} & \text{for all } i < j. \end{array} \tag{1}$$

The objective was first presented for signed graphs, by Bansal et al. [9] and by Shamir et al. [46]. Since its introduction, numerous variations on the objective have been presented for different weighted cases and graph types [2, 3, 14, 17, 34, 42, 54]. In *bipartite correlation clustering* [2, 5, 8, 15], nodes can be organized into two different sets, in such a way that $w_{ij}^+ = w_{ij}^- = 0$ for any pair of nodes i and j in the same set. In the complete, unweighted bipartite signed graph case, the best approximation factor proven is 3 [15].

2.2 Graph Clustering

Graph clustering is the task of separating the nodes of a graph into clusters in such a way that nodes inside a cluster share many edges with each other, but few with the rest of the graph. For an overview of graph clustering and community detection, we refer to surveys by Fortunato and Hric [19], and Schaeffer [45]. Given a graph $G = (V, E)$, we let $C = \{S_1, S_2, \ldots, S_k\}$ represent a disjoint clustering of V, with $S_i \cap S_j = \emptyset$ for $i \neq j$, and $\bigcup_i S_i = V$. Given a set of nodes $S \subseteq V$, let $\overline{S} = V \backslash S$ denote the complement set, and $\mathbf{cut}(S)$ be the weight of edges between S and $\overline{S}$. One of the most common approaches to graph clustering is to set up and solve (or approximate) a combinatorial objective function that encodes some notion of clustering structure. One common objective used for bipartitioning a graph is the normalized cut objective, defined for a set $S \subseteq V$ to be

$$\phi(S) = \frac{\mathbf{cut}(S)}{\mathbf{vol}(S)} + \frac{\mathbf{cut}(S)}{\mathbf{vol}(\overline{S})}, \tag{2}$$

where $\mathbf{vol}(S) = \sum_{i \in S} d_i$, with d_i being the degree of node i. Another very popular approach is to maximize the modularity objective [39], which measures the difference between the number of edges inside a cluster, and the expected number of edges in the cluster, where expectation is defined by some underlying graph null model.

Flexible parametric frameworks for graph clustering. Recently, we introduced a framework for graph clustering based on correlation clustering called LambdaCC [54]. Given a graph $G = (V, E)$, the LambdaCC framework replaces an edge $(i, j) \in E$ with a positive edge of weight $1 - \lambda d_i d_j$. For every pair $(i, j) \notin E$, a negative edge

of weight $\lambda d_i d_j$ is introduced. The resulting signed graph can then be partitioned with respect to the correlation clustering objective. LambdaCC generalizes several other objectives including normalized cut [47], modularity [39], and cluster deletion [46].

2.3 Hypergraph clustering

We let $\mathcal{H} = (V, \mathcal{E})$ denote a hypergraph, where V is a set of nodes, and $\mathcal{E}$ is a set of *hyperedges*, which involve two or more nodes. In hypergraphs, the notion of cuts and clustering becomes even more complex, as there can be numerous ways to partition the nodes of a hyperedge, and numerous ways to generalize a graph-based objective. We say that a hyperedge $e \in \mathcal{E}$ is *cut* if it spans at least two clusters of a clustering, C. In many clustering applications, any way of separating the nodes of a hyperedge is associated with a penalty equal to the weight of the hyperedge, though other more general notions of hyperedge cuts have also been considered [13, 22, 35, 36]. Given a set of nodes $S \subseteq V$ in a hypergraph $\mathcal{H}$, we let $\partial S = \{e \in \mathcal{E} : S \cap e \neq \emptyset, \bar{S} \cap e \neq \emptyset\}$ denote the boundary of S, and use $\mathbf{cut}_{\mathcal{H}}(S)$ to denote the hypergraph cut penalty for S. The most basic type of cut penalty is to simply count the number of edges on the boundary: $\mathbf{cut}_{\mathcal{H}}(S) = |\partial S|$. In this paper we also will consider the *linear* cut penalty, defined as follows:

$$\mathbf{cut}_{\mathcal{H}}(S) = \textstyle\sum_{e \in \mathcal{E}} \min\{|S \cap e|, |\bar{S} \cap e|\}\,. \tag{3}$$

Hypergraph generalizations of the normalized cut objective have also been introduced in practice [35, 36, 60]. Here we consider the following definition, first introduced for generalized hypergraph cut functions by Li et al. [35]:

$$\phi_{\mathcal{H}}(S) = \frac{\mathbf{cut}_{\mathcal{H}}(S)}{\mathbf{vol}_{\mathcal{H}}(S)} + \frac{\mathbf{cut}_{\mathcal{H}}(S)}{\mathbf{vol}_{\mathcal{H}}(\bar{S})}\,, \tag{4}$$

where $\mathbf{cut}_{\mathcal{H}}$ is any hypergraph cut function (e.g., $|\partial S|$ or (3)), and $\mathbf{vol}_{\mathcal{H}}(S) = \sum_{s \in S} d_s$ is the hypergraph volume of S. In this paper we will always consider the hypergraph degree d_s of a node to be the number of hyperedges a node participates in, though other definitions are possible [35, 36]. We also note that hypergraph generalizations of the modularity objective have been considered in different contexts [27, 32].

3 PARAMETRIC HYPERGRAPH CLUSTERING

Our first contribution is a hypergraph clustering objective that differentially treats hyperedges and pairwise edges in a parametric fashion. We further develop equivalence results with existing fixed-parameter objectives; algorithms are discussed in Section 5. Given a hypergraph $\mathcal{H} = (V, \mathcal{E})$ and a resolution parameter $\lambda \in (0, 1)$, we introduce a negative edge between each pair of nodes $(i, j) \in V \times V$, with weight $\lambda w_i w_j$, where w_i is a weight associated with node i. We consider either unit node weights ($w_i = 1$ for all nodes), or degree-based weights: $w_i = d_i$ for each $i \in V$. We treat each original hyperedge in $\mathcal{H}$ as a positive edge of weight 1. In order to accommodate a broad range of possible hyperedge cut penalties, we use the following general abstraction: let P_V be the family of all clusterings, and define $\zeta : \mathcal{E} \times P_V \to \mathbb{R}$ to be a function that outputs a penalty for the way in which clustering $C \in P_V$ separates the nodes of a hyperedge $e \in \mathcal{E}$. The HyperLam objective for a clustering C of $\mathcal{H}$ is then:

$$\textsc{HyperLam}(C, \lambda) = \textstyle\sum_{e \in \mathcal{E}} \zeta(e, C) + \sum_{i<j} \lambda w_i w_j (1 - z_{ij})\,. \tag{5}$$

where z_{ij} is a binary indicator for whether nodes i and j are separated ($z_{ij} = 1$) or clustered together ($z_{ij} = 0$) in C. This objective is inspired by the parametric LambdaCC objective for graphs [54].

In practice, there may be many meaningful cut functions ζ to consider—here we focus mostly on two. The first is the standard *all-or-nothing* penalty, typically considered in the higher-order correlation clustering literature, which assigns a penalty proportional to the weight of the hyperedge if and only if the hyperedge is cut (at least two of its nodes are separated). Formally, this is defined as

$$\zeta(e, C) = \begin{cases} 0 & \text{if } e \subseteq S \text{ for some } S \in C\,, \\ 1 & \text{otherwise.} \end{cases} \tag{6}$$

When this standard cut penalty is applied, objective (5) can be viewed as an instance of higher-order correlation clustering [20, 21, 30, 34] with a very special type of negative hyperedge set. Namely, there are no negative hyperedges of size three or more, but *every* pair of nodes defines a negative hyperedge of size two (i.e., a negative edge). The other cut function we consider is a multiway generalization of the linear hypergraph cut penalty (3), defined by

$$\zeta(e, C) = |e| - \max_{S \in C} |e \cap S|\,. \tag{7}$$

Given a clustering C, this function assigns a penalty equal to the minimum number of nodes of a hyperedge e that must be moved in order for e to be contained in a single cluster.

Given any hyperedge cut function ζ, the goal is to optimize (5) over all possible clusterings of nodes V. Our first theoretical result is to show that our new objective captures a hypergraph generalization of normalized cut [35], just as the LambdaCC graph clustering framework generalizes normalized cut [54]. With unit node weights ($w_i = 1$ for all i), Theorem 3.1 becomes a statement about a hypergraph variant of the sparsest cut clustering objective.

Theorem 3.1. *For degree-weighted HyperLam, there exists some $\lambda \in (0, 1)$, such that optimizing* (5) *over biclusterings of the form $C = \{S, \bar{S}\}$ for some $S \subseteq V$, will produce the minimum hypergraph normalized cut partition* (4)*. Furthermore, if the linear penalty* (7) *is used and we optimize over an arbitrary number of clusters, there exists some λ' such that* (5) *will be minimized by the minimum hypergraph normalized cut objective under the linear hypergraph cut function* (3)*.*

A proof is included in a full version of the manuscript [57].

4 PARAMETRIC BIPARTITE CLUSTERING

Next, we present a parameterized variant of bipartite correlation clustering in graphs, which we prove generalizes a number of other bipartite graph clustering, and comes with several novel approximation guarantees. Let $G = (V_1, V_2, E)$ be a bipartite graph in which V_1 and V_2 are node sets and E is a set of edges between nodes in V_1 and V_2. In order to define an instance of Parametric Bipartite Correlation Clustering (PBCC), we first define parameters μ_1, μ_2, and β, all in the interval $[0, 1]$. We then associate each $e \in E$ with a positive edge of weight $1 - \beta$, and every $e \in (V_1 \times V_2) - \{E\}$ with a negative edge with weight β. Additionally, each pair of nodes in V_1 is given a negative edge of weight μ_1, and each pair of nodes in V_2 is given a negative edge of weight μ_2. The result is a complete, weighted

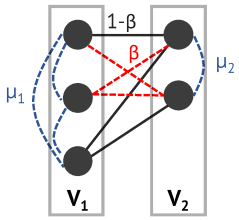


Figure 1: Parameterized BCC is given by a complete signed graph with edge weights parameterized by μ_1, μ_2 and β. Edges of weight β correspond to missing edges in some underlying bipartite graph $G = (V_1, V_2, E)$.

Table 1: Equivalence and approximation results for PBCC; ε represents a small, graph dependent number.

Parameters	Equivalence	Approx.
$\beta = \mu_1 = \mu_2 = \lambda$	LambdaCC	see [54], [21]
$\mu_1 = \mu_2 \geq (1-\beta)$	Bipart. Matching	1 (Thm 4.1)
$\mu_1 = \mu_2 = 0, \beta \geq 1-\varepsilon$	Bicluster deletion	4 (Thm 5.2)
$\mu_1 = \mu_2 = 0, \beta \geq \frac{1}{2}$	Generalized BCC	$6 - \frac{1}{\beta}$ (Thm 5.3)
$\mu_1 = \mu_2 \in [0,1], \beta \geq \frac{1}{2}$	-	5 (Thm 5.4)
$\mu_1 = \lambda, \mu_2 = 0, \beta = 0$	HyperLam	$O(\log n)$

instance of correlation clustering, where the underlying positive edge structure is a bipartite graph. We illustrate an instance of the problem in Figure 1. Our PBCC objective is

$$\begin{aligned}\text{PBCC}(C) =& \textstyle\sum_{i\in V_1,\, j\in V_2}[\beta(1-A_{ij})(1-z_{ij}) + (1-\beta)A_{ij}z_{ij}] \\ &+ \textstyle\sum_{(i,j)\in V_1\times V_1} \mu_1(1-z_{ij}) + \sum_{(i,j)\in V_2\times V_2} \mu_2(1-z_{ij}),\end{aligned} \tag{8}$$

where $A_{ij} = 1$ if $(i,j) \in E$, but is zero otherwise, and z_{ij} is the indicator for node separation in C ($z_{ij} = 0$ means (i,j) are clustered together) as before. Objective (8) generalizes several other well-studied problems. When $\mu_1 = \mu_2 = 0$ and $\beta = 1/2$, the problem corresponds to the standard unweighted bipartite correlation clustering problem (BCC) [2, 5]. When $\mu_1 = \mu_2 = \beta$, it is equivalent to applying the LambdaCC framework [54] to a bipartite graph. Table 1 summarizes other equivalence results and approximations for PBCC.

If $\beta > |V_1||V_2|/(|V_1||V_2| + 1)$ and $\mu_1 = \mu_2 = 0$, then making a mistake at a single negative edge of weight β introduces a greater weight of disagreements than placing each node into a singleton cluster. Therefore, the objective will be optimized by making a minimum number of positive-edge mistakes, subject to all clusters being bicliques. Thus, in this parameter regime, PBCC is equivalent to bicluster deletion, the problem of removing a minimum number of edges from a bipartite graph to partition it into disjoint bicliques.

Although PBCC is NP-hard in general, our next theorem, proven in the extended version [57], shows that in a certain parameter regime, PBCC is equivalent to solving a bipartite matching problem on $G = (V_1, V_2, E)$. Therefore, the problem can be solved in polynomial time in this regime.

Algorithm 1 Pivot

Input: Unweighted signed graph $G = (V, E^+, E^-)$
Output: Clustering C = Pivot(G)
Select a pivot node $k \in V$
Form cluster $S = \{v \in V : (k,v) \in E^+\} \cup \{v\}$
Output clustering $C = \{S, \text{Pivot}(G \backslash S)\}$

Algorithm 2 GenRound

Input: CC instance $G = (V, W^+, W^-)$, parameter $\delta \in [0,1]$.
Output: Clustering C of G.
Solve LP-relaxation of (1) to obtain *distances* x_{ij}, for each $i \neq j$.
$\tilde{E}^+ \leftarrow \{(i,j) : x_{ij} < \delta\}$, $\tilde{E}^- \leftarrow \{(i,j) : x_{ij} \geq \delta\}$
Apply Pivot to $\tilde{G} = (V, \tilde{E}^+, \tilde{E}^-)$.

Theorem 4.1. *If parameters μ_1, μ_2, and β satisfy $\min\{\mu_1, \mu_2\} \geq (1-\beta)$, then the optimal solution to PBCC for these parameters is the same as finding a maximum bipartite matching on $G = (V_1, V_2, E)$.*

When $\mu_1 = \lambda$, $\mu_2 = 0$, and $\beta = 0$, PBCC is equivalent to a special instance of HyperLam with a linear hyperedge cut penalty (7). Consider applying the HyperLam objective to a hypergraph $\mathcal{H} = (V, \mathcal{E})$ with unit node weights: $w_i = 1$ for all $i \in V$. When we use the linear hyperedge cut penalty (7), the HyperLam objective is equivalent to an instance of correlation clustering defined by performing a star expansion [61]. This replaces each hyperedge e with an auxiliary node v_e, and links every node in e to v_e with a unit-weight edge. This results in an instance of PBCC where $V_1 = V$ is the set of original nodes, each pair of which has a negative edge of weight $\mu_1 = \lambda$. The auxiliary nodes constitute V_2, with $\mu_2 = 0$, and edges between V and V_2 all have weight $1 - \beta = 1$.

5 APPROXIMATIONS AND HEURISTICS

We now turn our attention to approximation guarantees that can be obtained for our objectives in different parameter regimes. We begin by reviewing a general strategy for approximating variants of correlation clustering, through which we prove approximation guarantees for PBCC. In order to approximate HyperLam, we combine existing approximation algorithms for correlation clustering with techniques for reducing a hypergraph to a related pairwise graph. We conclude with heuristic approaches for HyperLam.

5.1 General LP Rounding Algorithm for CC

Pivot (aka Algorithm 1) is a simple algorithm for unweighted correlation clustering. When pivots are chosen uniformly at random, Ailon et al. [3] showed that this algorithm returns a 3-approximation for complete unweighted correlation clustering. Later, van Zuylen and Williamson [52] produced a de-randomized 3-approximation. Derived from this is a generic approximation algorithm scheme for new parametric correlation clustering variants. Pseudocode for this method, which we call GenRound, is given in Algorithm 2.

Theorem 5.1. *(Theorem 3.1 in [52]). Given a weighted instance of correlation clustering $G = (V, W^+, W^-)$, let $c_{ij} = w^+_{ij}x_{ij} + w^-_{ij}(1 - x_{ij})$. GenRound returns an α-approximation for the min-disagree*

objective (1) *if the threshold parameter, δ, is chosen so that the graph $\tilde{G} = (V, \tilde{E}^+, \tilde{E}^-)$ satisfies the following conditions:*

(1) For all $(i,j) \in \tilde{E}^+$, we have $w^-_{ij} \le \alpha c_{ij}$, and for all $(i,j) \in \tilde{E}^-$, we have $w^+_{ij} \le \alpha c_{ij}$.

(2) For every triangle (i,j,k) in $\tilde{G}$, with $\{(i,j),(j,k)\} \subseteq \tilde{E}^+$ and $(i,k) \in \tilde{E}^-$, we have $w^+_{ij} + w^+_{jk} + w^-_{ik} \le \alpha \left(c_{ij} + c_{jk} + c_{ik}\right)$.

When applying Pivot in Algorithm 2, selecting the pivot node uniformly at random gives an *expected* α-approximation. A deterministic algorithm with the same approximation factor α can be obtained via a careful selection of pivot nodes [52].

5.2 Graph Reductions for HyperLam

Although HyperLam is NP-hard to optimize, we can obtain approximation algorithms for the objective using two different techniques for converting hypergraphs to graphs.

Weighted clique expansion: Replace each hyperedge $e \in \mathcal{E}$ with a clique on e where each edge has weight $1/(|e|-1)$. If two nodes appear together in multiple hyperedges, assign a weight equal to the sum of weights from each such clique expansion.

Star expansion: Replace each hyperedge $e \in \mathcal{E}$ by adding an auxiliary node v_e and linking v_e to every node in e with a (positive) edge of weight 1. If we use weights $w_i = 1$ for all $i \in V$, this is equivalent to an instance of PBCC with $\mu_1 = \lambda$, $\mu_2 = 0$, and $\beta = 0$.

For each expansion technique, there is a negative edge of weight $\lambda w_i w_j$ between each pair $(i,j) \in V \times V$, where w_i is the weight for node i. The result is an instance of weighted correlation clustering that can be approximated with existing algorithms.

The weighting scheme for the clique expansion is chosen specifically to approximately model the all-or-nothing hyperedge cut penalty (6). For three-uniform hypergraphs, the relationship is exact [25]. For a k-node hyperedge, with $k > 3$, the minimum penalty for splitting the clique comes from placing all but one node in the same cluster, giving a penalty equal to $(k-1)/(k-1) = 1$. The maximum possible penalty, when all k nodes in e are placed in different clusters, is $\binom{k}{2}\frac{1}{k-1} = \frac{k}{2}$. Thus, the penalty at each positive hyperedge in the resulting reduced graph will be within a factor $k/2$ of the original all-or-nothing penalty for any clustering C.

Meanwhile, the star expansion enables us to *exactly* model the linear cut penalty (7). Each auxiliary node v_e is attached only to nodes that define a hyperedge e in the original hypergraph. Therefore, in the optimal clustering of the star expansion graph, v_e will be placed in the cluster that has the most nodes from e. The penalty then will equal the number of nodes that are clustered away from v_e, which is exactly the linear penalty (7). Thus, applying existing algorithms for correlation clustering [14, 17], we get an $O(k \log n)$ approximation for HyperLam with all-or-nothing penalty via the weighted clique expansion, and an $O(\log n)$ approximation for HyperLam with linear hyperedge penalty via the star expansion.

5.3 A Four-Approx for Bicluster Deletion

We now show how GenRound and Theorem 5.1 combine to develop a 4-approximation for bicluster deletion: the first constant-factor approximation for this problem. Rather than the edge weights presented in the last section, we view bicluster deletion as a general weighted correlation clustering problem with the following weights

$$(w^+_{ij}, w^-_{ij}) = \begin{cases} (0,0) & \text{if } i \text{ and } j \text{ are in the same bipartition of } G \\ (1,0) & \text{if } (i,j) \in E^+ \\ (0,\infty) & \text{if } (i,j) \in E^-. \end{cases}$$

Above, E^+ and E^- denote positive and negative edges between the two sides of the bipartite graph. To ensure no mistakes are made at negative edges, we add the constraint $x_{ij} = 1$ to BLP (1), for every $(i,j) \in E^-$. The LP-relaxation of this problem is given by

$$\begin{array}{lll} \text{minimize} & \sum_{(i,j)\in E^+} x_{ij} & \\ \text{subject to} & x_{ij} = 1 & \text{for all } (i,j) \in E^- \\ & x_{ij} \le x_{ik} + x_{jk} & \text{for all } i,j,k \\ & 0 \le x_{ij} \le 1 & \text{for all } i < j. \end{array} \tag{9}$$

Theorem 5.2. *Applying GenRound to LP* (9)*, with $\delta = 1/2$, returns a* 4*-approximation to bicluster deletion.*

The proof, included in the full version [53], relies on verifying that the conditions of Theorem (5.1) hold with $\alpha = 4$.

5.4 Generalized Results for PBCC

We now turn to approximation algorithms for a wider range of parameter settings. In the remainder of the section, we specifically consider $\mu = \mu_1 = \mu_2$. As we did for bicluster deletion, our goal is to find a threshold parameter δ and an approximation factor α such that the two conditions of Theorem 5.1 hold. We defer proofs to the full version of the manuscript [57].

Ailon et al. [2] proved a 4-approximation for unweighted bipartite correlation clustering, which is equivalent to PBCC with $\mu = 0$ and $\beta = 1/2$. We show how to select δ in GenRound so that not only can we recover this same approximation guarantee when $\mu = 0$ and $\beta = 1/2$, but also obtain guarantees for all $\beta \in \left[\frac{1}{2}, 1\right)$.

Theorem 5.3. *When $\mu = \mu_1 = \mu_2 = 0$ and $\beta \ge \frac{1}{2}$, Algorithm 2 with $\delta = 2\beta/(6\beta - 1)$ returns a $(6 - 1/\beta)$-approximation for PBCC.*

Considering a more general parameter regime, where $\mu_1 = \mu_2 \in [0,1]$, we obtain a 5-approximation for all $\beta \ge 1/2$.

Theorem 5.4. *When $\mu_1 = \mu_2$ and $\beta \ge \frac{1}{2}$, Algorithm 2 with $\delta = 2/5$ returns a* 5*-approximation to PBCC.*

5.5 Modularity Connections and Heuristics

Returning to the HyperLam objective, applying our weighted clique expansion and introducing a negative edge of weight $\lambda d_i d_j$ for node pair (i,j) is equivalent to solving a weighted variant of the LambdaCC graph clustering objective [54]. Since LambdaCC is equivalent to a generalization of modularity with a resolution parameter [39, 54], we can also approximately optimize the HyperLam objective by applying our weighted clique expansion and then running heuristic algorithms for modularity such as the Louvain algorithm [11] or, more appropriately, generalizations of Louvain with a resolution parameter [26]. A similar approach will also work for the star expansion: we set the weight of a node in V to be its hyperedge degree $w_v = d_v$, and the weight of an auxiliary node v_e (obtained from expanding a hyperedge) to be $w_{v_e} = 0$. This also corresponds to a weighted variant of LambdaCC, since each pair of nodes (i,j) in the graph share a negative edge of weight $\lambda w_i w_j$. In

many cases this weight will be zero, but we can still apply generalized Louvain-style heuristics to optimize the objective.

Kumar et al. [32] previously considered a modularity-based approach for hypergraph clustering based on the same type of clique expansion. These authors applied the same weight, $1/(|e|-1)$, to each edge in a clique expansion of a hypergraph $|e|$, as this preserves the degree distribution of nodes in the original hypergraph. They then considered applying the modularity objective [39] to the resulting graph. Their approach corresponds to applying a weighted clique expansion to an instance of HyperLam, and setting $\lambda = 1/(\mathbf{vol}_{\mathcal{H}}(V))$. Thus, this approach can be viewed as a special case of our hyperedge expansion procedure for HyperLam. The connection to correlation clustering we show, along with the resulting approximation algorithms for the all-or-nothing hypergraph cut, provide further theoretical motivation for this choice of weighted clique expansion. Despite this connection to a previous clique expansion technique for hypergraph modularity, we note that our original hypergraph objective (5) nevertheless differs from generalizations of modularity defined directly for hypergraphs [27], as opposed to modularity objectives applied to clique expansions of hypergraphs.

6 RELATED WORK

To anchor our work, we highlight related results on algorithms for correlation clustering, techniques for parametric clustering in standard graphs, and recent results on clustering hypergraphs.

Correlation Clustering Bansal et al. [9] first introduced the problem of correlation clustering, providing a constant factor approximation for the complete unweighted case. Amit was the first to consider the problem in the bipartite setting [5], providing an 11-approximation for the complete unweighted setting. Later, Ailon et al. [2] presented a 4-approximation. Most recently, Chawla et al. [15] improved the best approximation factor to 3.

Higher-order correlation clustering was first considered by Kim et al. [30] in the content of image segmentation. Li et al. [34] were the first to develop approximation algorithms for the complete 3-uniform case, giving a 9-approximation. We later gave a $4(k-1)$ approximation for the k-uniform setting, which was then improved to $2k$ by Li et al. [37]. For weighted hypergraphs, Fukunaga [20] presented an $O(k \log n)$ approximation algorithm, where k is the maximum size of negative hyperedges.

Parametric Graph Clustering Our introduction of the LambdaCC framework situates graph clustering within correlation clustering [54]. We proved equivalence results with modularity, normalized cut, and sparsest cut, and gave a 3-approximation when $\lambda \geq 1/2$, based on LP-rounding. We were later able to show that the LP relaxation has an integrality gap of $O(\log n)$ for some small values of λ [21]. LambdaCC is in turn related to other graph parametric clustering objectives, such as stability [16], various Potts models [43, 50], and generalizations of modularity [7].

Hypergraph Clustering Different higher-order generalizations of modularity have been previously developed [27, 32], along with higher-order variants of conductance [10] and normalized cut [35, 60]. In hypergraph clustering, the most common penalty for a cut hyperedge is the weight of that hyperedge, regardless of how the hyperedge is cut. However, other penalties have also been considered in the context of hypergraph partitioning and clustering [13, 35, 36]. A more comprehensive overview of generalized hypergraph cut functions is included in recent work by one of the authors [53].

7 EXPERIMENTS

We demonstrate our parametric objectives and algorithms in analyzing an assortment of different types of datasets. Our primary goal is to highlight the diversity of results we can achieve. We begin by running our approximation algorithms for PBCC on several bipartite datasets to illustrate the algorithmic performance and output in different parameter regimes. We then apply the HyperLam framework to motif clustering. Finally, we apply our framework to detect product categories in an Amazon product review hypergraph.

Implementation Details. We implement our algorithms in Julia, using Gurobi to solve LP relaxations. Code for all algorithms and experiments are available online at https://github.com/nveldt/ParamCC. We focus on studying the differences among the objective functions rather than optimizing implementations. Our motif clustering experiments were run on a laptop with 8GB of RAM. All other experiments were run on a larger machine with four 16-core Intel Xeon E7-8867 v3 processors. Running large instances with Louvain-style algorithms was not a bottleneck and these always finished in a few minutes or less. On the bipartite graphs we consider, running our PBCC algorithms typically took a few seconds or a few minutes. Solving the correlation clustering LP relaxation for larger graphs is often very expensive; this is, however, an active research area [12, 44, 48, 56] and solvers have been produced for around 20,000-node graphs. This leaves us with a theory/practice gap between the effective Louvain-based heuristics and more principled approximations that we intend to study in the future.

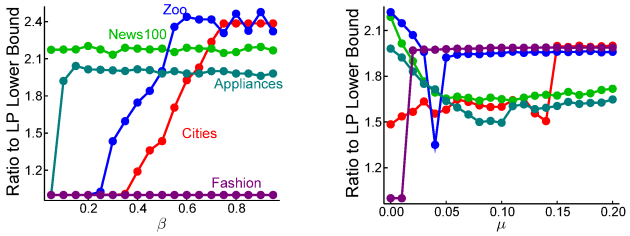
7.1 PBCC on Real Bipartite Graphs

We run our PBCC approximation algorithms on five bipartite graphs constructed from real data[1], with a range of parameter settings.

- The *Cities* graph encodes which set of 46 global firms (nodes on side V_1) have offices in 55 different major cities (nodes on side V_2).
- *Newgroups100* is made up of a set of 100 documents (V_1) and 100 words (V_2); edges indicate words used in each document. We have extracted a random subset of 100 documents (25 from each of four categories: *sci*∗, *comp*∗, *rec*∗, and *talk*∗) from a larger dataset, often used as a benchmark for *hypergraph* clustering [24, 36, 60].
- The *Zoo* dataset encodes 100 animals and their associations with 15 different binary attributes (e.g., "hair", "feathers", "eggs").
- The last two bipartite graphs are constructed from reviewers on Amazon (V_1) that have reviewed products (V_2) within certain categories [40]. The *Fashion* category has 404 reviewers and 31 products, and *Appliances* has 44 reviewers for 48 products.

Figure 2 displays a posteriori approximation ratios for our method (objective score divided by LP lower bound), first for $\mu_1 = \mu_2 = 0$ and $\beta \in [0, 1]$, and then for $\beta = 1/2$ and $\mu = \mu_1 = \mu_2 \in [0, 0.2]$. After solving the LP relaxation for each (μ, β) pair, we try rounding with δ values from 0.05 to 0.95 in increments of 0.05, taking the result with the best objective score, since the rounding procedure is much

[1] Cities: https://www.lboro.ac.uk/gawc/datasets/da6.html; Newsgroups: www.cs.nyu.edu/~roweis/data/; Zoo: https://archive.ics.uci.edu/ml/datasets/zoo. Amazon (5-core): https://nijianmo.github.io/amazon/index.html.



(a) $\mu = 0, 0 \leq \beta \leq 1$ (b) $\beta = 0.5, 0 \leq \mu \leq 0.2$

Figure 2: A posteriori approximation ratios for running our LP-based PBCC algorithms on real-world bipartite graphs.

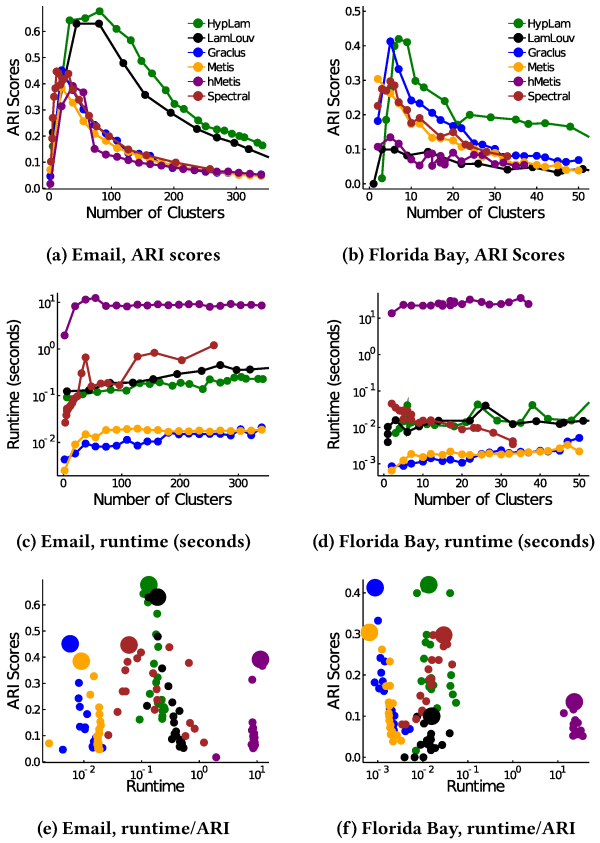
faster than the initial LP solve. We note that the approximation factor curve varies significantly from dataset to dataset. However, in all cases we obtain much better approximation factors than the ones given in Table 1, even for β values where our algorithms have no formal guarantees. In certain regimes we also observe abrupt changes in approximation factors, e.g., for *Fashion* when $\beta = 0.5$ and μ is near zero (Figure 2b). We also tested $\mu > 0.2$ when $\beta = 0.5$. In this parameter regime, the problem is nearly the same as bipartite matching, though our LP-based approach only provides a posteriori guarantees of around a factor 2. This motivates the question of what other approximation algorithms might perform better when the problem is "almost" bipartite matching.

7.2 HyperLam for Motif Clustering

HyperLam can detect motif-rich clusters at different resolutions in a graph. In motif clustering, a small, frequently repeated subgraph (a motif) is identified, and each motif instance is associated with a hyperedge [6, 10, 35, 51]. Applying a hypergraph clustering technique penalizes the number of cut motifs, rather than just the cut edges. This encourages keeping whole motifs inside clusters.

Triangles are known to be important motifs for identifying community structure in networks [31, 51]. We therefore apply the HyperLam framework to cluster the Email-EU dataset [33, 59] based on triangles. Each edge in the graph (which we treat as undirected) represents an email sent between members of a European research institution. A metadata label indicating each researcher's department comes with each node.

To find clusters at different resolutions in the graph, we approximate the HyperLam objective by first applying a clique expansion based on triangle motifs. Since the motif has three nodes, the all-or-nothing cut is the same as the linear penalty, and the clique expansion perfectly models both. We cluster the resulting *weighted* graph with a weighted version of Lambda-Louvain [54], which makes greedy local node moves similar to the Louvain method [11], but optimizes a different objective. We compare against running Lambda-Louvain on the original graph. We also compare against standard graph algorithms Metis [28] and Graclus [18], varying the number of clusters k, and recursive spectral partitioning, for a range of different minimum cluster sizes m_{size}. We test these last three methods on both the original graph and clique-expanded graph, but show results only for the clique-expanded graph, as this leads to the best outcome for these methods. Finally, we run



(a) Email, ARI scores (b) Florida Bay, ARI Scores

(c) Email, runtime (seconds) (d) Florida Bay, runtime (seconds)

(e) Email, runtime/ARI (f) Florida Bay, runtime/ARI

Figure 3: (a) HyperLam with triangle motifs better captures the relationship between community structure and department labels of researchers at a European research institution, across all clusters sizes. (b) Optimizing HyperLam using the *bifan* motif and the inhomogeneous hyperedge splitting function of Li et al. [35], we find clusterings with higher correlation with biological classifications of species in a food web. Figures (c) and (d) display runtimes, while (e) and (f) display the trade-off between runtime and ARI score. Larger dots mark the best ARI score for each method.

hMetis (a hypergraph variant of Metis) on the hypergraph formed by associating motifs with hyperedges, varying cluster number, k.

After forming multiple clusterings with each method for many parameter values (k, m_{size}, or λ), we measure the Adjusted Rand Index score between each clustering and the known department metadata labels. Scores for each cluster size are displayed in Figure 3a. Although the department labels do not exactly match with community structure in the network, there is a strong correlation between the two, and the higher ARI scores obtained by running HyperLam with the triangle motif indicate that our method is best able to detect this relationship.

We perform a similar experiment on the Florida Bay food web, in which nodes indicate species (e.g., Isopods, Eels, Meroplankton), and directed edges indicating carbon exchange [10, 35]. Following the approach of Li and Milenkovic [35], we consider the *bifan* motif, in which two nodes $\{v_1, v_2\}$ have uni-directional edges to

two other nodes $\{v_3, v_4\}$, and any edge combination within sets $\{v_1, v_2\}$ and $\{v_3, v_4\}$ is allowed. We identify each instance of the motif as a hyperedge. Li and Milenkovic specifically use an *inhomogeneous* hyperedge cutting penalty, which can be modeled by simply adding undirected edges (v_1, v_2) and (v_3, v_4). Thus, we convert the input graph into a new graph, and cluster with a weighted version of Lambda-Louvain, to optimize the HyperLam objective. We again run hMetis on the hypergraph defined by motifs, and Lambda-Louvain on the undirected version of the original graph. We ran {Metis, Graclus, Recursive spectral} on the new graph obtained by expanding bifan motifs, as this led to better results than running them on the original graph. Figure 3b demonstrates that applying our HyperLam framework with the bifan motif structure leads to the highest ARI clustering scores with the biological classifications identified by Li et al. [35] (e.g. producers, fish, mammals). Acknowledging that our implementations are not optimized for speed, Figures 3e and 3f show that Metis, Graclus, and HyperLam methods constitute the efficient frontier.

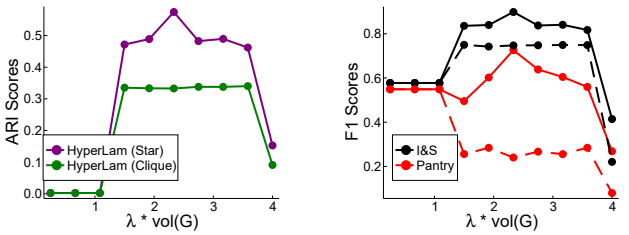
7.3 Clustering Amazon Products Categories

In our last experiment we illustrate differences that arise when applying the HyperLam framework with different hyperedge cut functions. In order to do so, we apply our framework to a hypergraph constructed from Amazon review data, similar to the *Fashion* and *Appliances* hypergraphs in the first experiment. This time, we extract nine product categories, associating each product in these categories with a node, and defining a hyperedge to be a set of all products that are reviewed by the same person. This results in a hypergraph with 13,156 nodes, 31,544 hyperedges, with the maximum and mean hyperedge sizes being 219 and 8.1, respectively. Each node is associated with exactly one category label.

As outlined in Section 5, we apply a weighted clique expansion and a star expansion to the Amazon review hypergraph, each modeling a different cut penalty. We scale the graphs so that they share the same total volume, then cluster them both with Lambda-Louvain, using various values of λ. Running Lambda-Louvain on the clique expansion took just over two minutes on average, while runtimes were just over four minutes on average for the star expansion.

The hypergraph has a single large connected component, indicating that reviewers do review products across different categories. At the same time, 95% of all hyperedges in the hypergraph are completely contained inside one of the sets of nodes defining a product category. Thus, we expect that clustering the hypergraph based on hyperedge structure will yield clusters that correlate highly with product categories. We confirm this by computing ARI scores between category labels and the clusterings returned by optimizing HyperLam for both graph expansions (Figure 4).

In order to better understand the structure of clusters formed by our methods, and their relationship with product categories, we measure how well each clustering detects individual product-category node sets in the hypergraph. For each category (e.g., "Appliances"), we measure how well a HyperLam clustering "tracks" that category by taking the best F1 score between any of the HyperLam clusters and the product-category node set in question. For example, if one of the clusters returned by HyperLam exactly matches the "Appliances" node set, then we have perfectly "tracked"



(a) ARI scores (b) Tracking categories

Figure 4: (a) The clique and star expansion lead to clusterings that are correlated with product categories in an Amazon product hypergraph. (b) We compute the best F1 score between clusters formed by HyperLam, and individual product category clusters. The star expansion (results with solid lines) is able to better track the two largest clusters, "Industrial and Scientific" (black) and "Prime Pantry" (red), compared to the clique expansion (dashed lines).

this category, and we report an F1 score of 1. Figure 4b illustrates that the star expansion is able to better track the two largest categories, "Prime Pantry" and "Industrial & Scientific", each of which has roughly 5000 nodes. This helps explain why the star expansion obtains higher ARI scores in general. On the other hand, we observed that the clique expansion tracks the "Software" category (802 nodes) better. This highlights the fact that different hyperedge cut functions can leads to substantially different types of clusters.

8 DISCUSSION

We have presented a new, flexible, and general framework for parametric clustering of hypergraph and bipartite graph datasets. This framework has deep connections to existing objective functions in the literature and there exist polynomial time approximation results as well as heuristic algorithms. While such frameworks are extremely useful to expert practitioners to engineer and investigate datasets, they are often challenging for less sophisticated users who have a tendency to rely on default parameters. Towards that end, there is a general need for statistical and automated techniques to help guide users to the most successful use of these methods, which is something we hope to design in the future.

Another challenge involves scaling of the parameters. In our experiments, we often scale these by the volume of the graph (the total sum of edge-weighted degrees) as that has proven to be successful in practice. However, it is unclear if this is the best approach in all circumstances, or whether in some situations the absolute values of the parameters should be preferred. Finally, as our experiments highlight, there are distinct phase transitions in the behavior among these different regimes; finding ways to identify these characteristic regions would also make these parametric objectives useful to automatically find characteristically different clusterings.

ACKNOWLEDGMENTS

This research was supported by NSF IIS-1546488, CCF-1909528, NSF Center for Science of Information STC, CCF-0939370, DOE DESC0014543, NASA, the Sloan Foundation, and the Melbourne School of Engineering.

REFERENCES

[1] Sameer Agarwal, Jongwoo Lim, Lihi Zelnik-Manor, Pietro Perona, David Kriegman, and Serge Belongie. 2005. Beyond Pairwise Clustering *(CVPR '05)*.

[2] Nir. Ailon, Noa. Avigdor-Elgrabli, Edo. Liberty, and Anke. van Zuylen. 2012. Improved Approximation Algorithms for Bipartite Correlation Clustering. *SIAM J. Comput.* 41, 5 (2012), 1110–1121.

[3] Nir Ailon, Moses Charikar, and Alantha Newman. 2008. Aggregating inconsistent information: ranking and clustering. *Journal of the ACM (JACM)* 55, 5 (2008), 23.

[4] Ilya Amburg, Nate Veldt, and Austin R Benson. Clustering in graphs and hypergraphs with categorical edge labels *(WWW '20)*.

[5] Noga Amit. 2004. *The bicluster graph editing problem.* Master's thesis. Tel Aviv University.

[6] A Arenas, A Fernández, S Fortunato, and S Gómez. 2008. Motif-based communities in complex networks. *Journal of Physics A: Mathematical and Theoretical* 41, 22 (2008).

[7] A Arenas, A Fernández, and S Gómez. 2008. Analysis of the structure of complex networks at different resolution levels. *New Journal of Physics* 10, 5 (2008).

[8] M. Asteris, A. Kyrillidis, D. Papailiopoulos, and A. Dimakis. Bipartite correlation clustering: Maximizing agreements *(AISTATS '16)*.

[9] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. 2004. Correlation Clustering. *Machine Learning* 56 (2004), 89–113.

[10] Austin R. Benson, David F. Gleich, and Jure Leskovec. 2016. Higher-order organization of complex networks. *Science* 353, 6295 (2016), 163–166.

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

[12] Justin Brickell, Inderjit S. Dhillon, Suvrit Sra, and Joel A. Tropp. 2008. The Metric Nearness Problem. *SIAM J. Matrix Anal. Appl.* 30, 1 (2008), 375–396.

[13] Ümit V. Çatalyürek and Cevdet Aykanat. 1999. Hypergraph-Partitioning Based Decomposition for Parallel Sparse-Matrix Vector Multiplication. *IEEE Transactions on Parallel and Distributed Systems* 10, 7 (1999), 673–693.

[14] Moses Charikar, Venkatesan Guruswami, and Anthony Wirth. 2005. Clustering with qualitative information. *J. Comput. System Sci.* 71, 3 (2005), 360 – 383. Learning Theory 2003.

[15] Shuchi Chawla, Konstantin Makarychev, Tselil Schramm, and Grigory Yaroslavtsev. 2015. Near optimal LP rounding algorithm for correlation clustering on complete and complete k-partite graphs *(STOC '15)*. ACM.

[16] J.-C. Delvenne, S. N. Yaliraki, and M. Barahona. 2010. Stability of graph communities across time scales. *Proceedings of the National Academy of Sciences* 107, 29 (2010), 12755–12760.

[17] Erik D. Demaine, Dotan Emanuel, Amos Fiat, and Nicole Immorlica. 2006. Correlation clustering in general weighted graphs. *Theoretical Computer Science* 361, 2 (2006), 172 – 187. Approximation and Online Algorithms.

[18] Inderjit S. Dhillon, Yuqiang Guan, and Brian Kulis. 2007. Weighted Graph Cuts without Eigenvectors A Multilevel Approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29, 11 (2007), 1944–1957.

[19] Santo Fortunato and Marc Barthélemy. 2007. Resolution limit in community detection. *Proceedings of the National Academy of Sciences* 104, 1 (2007), 36–41.

[20] Takuro Fukunaga. 2018. LP-Based Pivoting Algorithm for Higher-Order Correlation Clustering. In *Computing and Combinatorics*.

[21] David F. Gleich, Nate Veldt, and Anthony Wirth. 2018. Correlation Clustering Generalized *(ISAAC 2018)*.

[22] J. Gong and Sung Kyu Lim. 1998. Multiway partitioning with pairwise movement *(ICAD '98)*.

[23] S. W. Hadley, B. L. Mark, and A. Vannelli. 1992. An efficient eigenvector approach for finding netlist partitions. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 11, 7 (1992).

[24] Matthias Hein, Simon Setzer, Leonardo Jost, and Syama Sundar Rangapuram. 2013. The Total Variation on Hypergraphs - Learning on Hypergraphs Revisited *(NIPS'13)*.

[25] Edmund Ihler, Dorothea Wagner, and Frank Wagner. 1993. Modeling Hypergraphs by Graphs with the Same Mincut Properties. *Inf. Process. Lett.* 45, 4 (1993).

[26] Lucas G. S. Jeub, Marya Bazzi, Inderjit S. Jutla, and Peter J. Mucha. 2011-2017. A generalized Louvain method for community detection implemented in MATLAB. (2011-2017). http://netwiki.amath.unc.edu/GenLouvain

[27] Bogumił Kamiński, Valérie Poulin, Paweł Prałat, Przemysław Szufel, and François Théberge. 2019. Clustering via hypergraph modularity. *PloS one* 14, 11 (2019).

[28] George Karypis and Vipin Kumar. 1998. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM J. Sci. Comput.* 20, 1 (1998), 359–392.

[29] George Karypis and Vipin Kumar. 1999. Multilevel K-way Hypergraph Partitioning *(DAC '99)*. ACM, 343–348.

[30] Sungwoong Kim, Sebastian Nowozin, Pushmeet Kohli, and Chang D. Yoo. 2011. Higher-Order Correlation Clustering for Image Segmentation *(NIPS '11)*.

[31] Christine Klymko, David F. Gleich, and Tamara G. Kolda. 2014. Using Triangles to Improve Community Detection in Directed Networks. In *The Second ASE International Conference on Big Data Science and Computing, BigDataScience.*

[32] Tarun Kumar, Sankaran Vaidyanathan, Harini Ananthapadmanabhan, Srinivasan Parthasarathy, and Balaraman Ravindran. 2020. A New Measure of Modularity in Hypergraphs: Theoretical Insights and Implications for Effective Clustering. In *Complex Networks and Their Applications VIII*. Springer International Publishing.

[33] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2007. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 1, 1 (2007), 2.

[34] Pan Li, H. Dau, Gregory J. Puleo, and Olgica Milenkovic. 2017. Motif clustering and overlapping clustering for social network analysis *(INFOCOM '17)*. 1–9.

[35] Pan Li and Olgica Milenkovic. 2017. Inhomogeneous Hypergraph Clustering with Applications *(NIPS '17)*. 2308–2318.

[36] Pan Li and Olgica Milenkovic. 2018. Submodular Hypergraphs: p-Laplacians, Cheeger Inequalities and Spectral Clustering *(ICML '18)*. 3020–3029.

[37] Pan Li, Gregory. J. Puleo, and Olgica. Milenkovic. 2019. Motif and Hypergraph Correlation Clustering. *IEEE Transactions on Information Theory* (2019), 1–1.

[38] Tom Michoel and Bruno Nachtergaele. 2012. Alignment and integration of complex networks by hypergraph-based spectral clustering. *Physical Review E* 86 (2012), 056111. Issue 5.

[39] Mark EJ Newman and Michelle Girvan. 2004. Finding and evaluating community structure in networks. *Physical review E* 69, 026113 (2004).

[40] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects *(EMNLP-IJCNLP '19)*. 188–197.

[41] Leto Peel, Daniel B. Larremore, and Aaron Clauset. 2017. The ground truth about metadata and community detection in networks. *Science Advances* 3, 5 (2017).

[42] Gregory. J. Puleo and Olgica. Milenkovic. 2018. Correlation Clustering and Biclustering With Locally Bounded Errors. *IEEE Transactions on Information Theory* 64, 6 (June 2018), 4105–4119.

[43] Jörg Reichardt and Stefan Bornholdt. 2004. Detecting Fuzzy Community Structures in Complex Networks with a Potts Model. *Phys. Rev. Lett.* 93 (2004), 218701.

[44] Cameron Ruggles, Nate Veldt, and David F. Gleich. A Parallel Projection Method for Metric Constrained Optimization *(SIAM CSC '20)*.

[45] Satu Elisa Schaeffer. 2007. Graph clustering. *Computer Science Review* (2007).

[46] Ron Shamir, Roded Sharan, and Dekel Tsur. 2004. Cluster graph modification problems. *Discrete Applied Mathematics* 144 (2004), 173–182.

[47] Jianbo Shi and J. Malik. 2000. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22, 8 (2000), 888–905.

[48] Rishi Sonthalia and Anna C. Gilbert. 2020. Project and Forget: Solving Large-Scale Metric Constrained Problems. (2020). arXiv:cs.LG/2005.03853

[49] Ze Tian, TaeHyun Hwang, and Rui Kuang. 2009. A hypergraph-based learning algorithm for classifying gene expression and arrayCGH data with prior knowledge. *Bioinformatics* 25, 21 (2009), 2831–2838.

[50] V. A. Traag, P. Van Dooren, and Y. Nesterov. 2011. Narrow scope for resolution-limit-free community detection. *Phys. Rev. E* 84 (Jul 2011), 016114. Issue 1.

[51] Charalampos E. Tsourakakis, Jakub Pachocki, and Michael Mitzenmacher. 2017. Scalable Motif-aware Graph Clustering *(WWW '17)*. 1451–1460.

[52] Anke van Zuylen and David P. Williamson. 2009. Deterministic Pivoting Algorithms for Constrained Ranking and Clustering Problems. *Mathematics of Operations Research* 34, 3 (2009), 594–620.

[53] Nate Veldt, Austin R. Benson, and Jon Kleinberg. 2020. Hypergraph Cuts with General Splitting Functions. (2020). arXiv:cs.DS/2001.02817

[54] Nate Veldt, David F. Gleich, and Anthony Wirth. 2018. A Correlation Clustering Framework for Community Detection *(WWW '18)*. 439–448.

[55] Nate Veldt, David F. Gleich, and Anthony Wirth. 2019. Learning Resolution Parameters for Graph Clustering *(WWW '19)*.

[56] Nate Veldt, David F. Gleich, Anthony Wirth, and James Saunderson. 2019. Metric-Constrained Optimization for Graph Clustering Algorithms. *SIAM Journal on Mathematics of Data Science* 1, 2 (2019), 333–355.

[57] Nate Veldt, Anthony Wirth, and David F. Gleich. 2020. Parameterized Correlation Clustering in Hypergraphs and Bipartite Graphs. (2020). arXiv:cs.DS/2002.09460

[58] Hao Yin, Austin R. Benson, and Jure Leskovec. 2018. Higher-order clustering in networks. *Phys. Rev. E* 97 (2018), 052306. Issue 5.

[59] Hao Yin, Austin R Benson, Jure Leskovec, and David F Gleich. 2017. Local higher-order graph clustering *(KDD '17)*. 555–564.

[60] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. 2006. Learning with Hypergraphs: Clustering, Classification, and Embedding *(NIPS '06)*.

[61] J. Y. Zien, M. D. F. Schlag, and P. K. Chan. 1999. Multilevel spectral hypergraph partitioning with arbitrary vertex sizes. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 18, 9 (1999), 1389–1399.