Clustering in graphs and hypergraphs with categorical edge labels

Ilya Amburg
Cornell University
ia244@cornell.edu

Nate Veldt
Cornell University
nveldt@cornell.edu

Austin R. Benson
Cornell University
arb@cs.cornell.edu

ABSTRACT

Modern graph or network datasets often contain rich structure that goes beyond simple pairwise connections between nodes. This calls for complex representations that can capture, for instance, edges of different types as well as so-called "higher-order interactions" that involve more than two nodes at a time. However, we have fewer rigorous methods that can provide insight from such representations. Here, we develop a computational framework for the problem of clustering hypergraphs with categorical edge labels — or different interaction types — where clusters corresponds to groups of nodes that frequently participate in the same type of interaction.

Our methodology is based on a combinatorial objective function that is related to correlation clustering on graphs but enables the design of much more efficient algorithms that also seamlessly generalize to hypergraphs. When there are only two label types, our objective can be optimized in polynomial time, using an algorithm based on minimum cuts. Minimizing our objective becomes NP-hard with more than two label types, but we develop fast approximation algorithms based on linear programming relaxations that have theoretical cluster quality guarantees. We demonstrate the efficacy of our algorithms and the scope of the model through problems in edge-label community detection, clustering with temporal data, and exploratory data analysis.

ACM Reference Format:
Ilya Amburg, Nate Veldt, and Austin R. Benson. 2020. Clustering in graphs and hypergraphs with categorical edge labels. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3366423.3380152

1 INTRODUCTION

Representing data as a graph or network appears in numerous application domains, including, for example, social network analysis, biological systems, the Web, and any discipline that focuses on modeling interactions between entities [5, 25, 48]. The simple model of nodes and edges provides a powerful and flexible abstraction, and over time, more expressive models have been developed to incorporate richer structure in data. In one direction, models now use more information about the nodes and edges: multilayer networks capture nodes and edges of different types [35, 47], metapaths formalize heterogeneous relational structure [24, 59], and graph convolutional networks use node features for prediction tasks [34]. In another direction, *group*, *higher-order*, or *multi-way* interactions between several nodes — as opposed to pairwise interactions — are paramount to the model. In this space, interaction data is modeled with hypergraphs [9, 65, 66], tensors [1, 7, 52], affiliation networks [39], simplicial complexes [10, 50, 53, 55], and motif representations [11, 54]. Designing methods that effectively analyze the richer structure encoded by these expressive models is an ongoing challenge in graph mining and machine learning.

In this work, we focus on the fundamental problem of clustering, where the general idea is to group nodes based on some similarity score. While graph clustering methods have a long history [26, 42, 46, 56], existing approaches for rich graph data do not naturally handle networks with categorical edge labels. In these settings, a categorical edge label encodes a type of discrete similarity score — two nodes connected by an edge with category label c are similar with respect to c. This structure arises in a variety of settings: brain regions are connected by different types of connectivity patterns [20]; edges in coauthorship networks are categorized by publication venues, and copurchasing data can contain information about the type of shopping trip. In the examples of coauthorship and copurchasing, the interactions are also higher-order — publications can involve multiple authors and purchases can be made up of several items. Thus, we would like a scalable approach to clustering nodes using a similarity score based on categorical edge labels that work well for higher-order interactions.

Here, we solve this problem with a novel clustering framework for edge-labeled graphs. Given a network with k edge labels (categories or colors), we create k clusters of nodes, each corresponding to one of the labels. As an objective function for cluster quality, we seek to simultaneously minimize two quantities: (i) the number of edges that cross cluster boundaries, and (ii) the number of intra-cluster "mistakes", where an edge of one category is placed inside the cluster corresponding to another category. This approach results in a clustering of nodes that respects both the coloring induced by the edge labels and the topology of the original network. We develop this computational framework in a way that seamlessly generalizes to the case of hypergraphs to model higher-order interactions, where hyperedges have categorical labels.

The style of our objective function is related to correlation clustering in signed networks [8], as well as its generalization for discrete labels (colors), chromatic correlation clustering [12, 13], which are based on similar notions of mistake minimization. However, a key difference is that our objective function does not penalize placing nodes not connected by an edge in the same cluster. This modeling difference provides serious advantages in terms of tractability, scalability, and the ability to generalize to higher-order interactions.

We first study the case of edge-labeled (edge-colored) graphs with only two categories. We develop an algorithm that optimizes

WWW '20, April 20–24, 2020, Taipei, Taiwan

ACM ISBN 978-1-4503-7023-3/20/04.
https://doi.org/10.1145/3366423.3380152

our Categorical Edge Clustering objective function in polynomial time by reducing the problem to a minimum s-t graph cut problem on a related network. We then generalize this construction to facilitate quickly finding the optimal solution exactly for hypergraphs. This is remarkable on two fronts. First, typical clustering objectives such as minimum bisection, ratio cut, normalized cut, and modularity are NP-hard to optimize even in the case of two clusters [17, 61]. And in correlation clustering, having two edge types is also NP-hard [8]. In contrast, our setup admits a simple algorithm based on minimum s-t cuts. Second, our approach seamlessly generalizes to hypergraphs. Importantly, we do not approximate hyperedge cuts with weighted graph cuts, which is a standard heuristic approach in hypergraph clustering [2, 44, 66]. Instead, our objective exactly models the number of hyperedges that cross cluster boundaries and the number of intra-cluster "mistake" hyperedges.

With more than two categories, we show that minimizing our objective is NP-hard, and we proceed to construct several approximation algorithms. The first set of algorithms are based on practical linear programming relaxations, achieving an approximation ratio of $\min\left\{2-\frac{1}{k}, 2-\frac{1}{r+1}\right\}$, where k is the number of categories and r is the maximum hyperedge size ($r = 2$ for the graph case). The second approach uses a reduction to multiway cut, where practical algorithms have a $\frac{r+1}{2}$ approximation ratio and algorithms of theoretical interest have a $2(1-\frac{1}{k})$ approximation ratio.

We test our methods on synthetic benchmarks as well as a variety of real-world datasets coming from neuroscience, biomedicine, and social and information networks; our methods work far better than baseline approaches at minimizing our objective function. Surprisingly, our linear programming relaxation often produces a rounded solution that matches the lower bound, i.e., it exactly minimizes our objective function. Furthermore, our algorithms are also fast in practice, often taking under 30 seconds on large hypergraphs.

We examine an application to a variant of the community detection problem where edge labels indicate that two nodes are in the same cluster and find that our approach accurately recovers ground truth clusters. We also show how our formulation can be used for temporal community detection, in which one clusters the graph based on topology and temporal consistency. In this case, we treat binned edge timestamps as categories, and our approach finds good clusters in terms of topological metrics *and* temporal aggregation metrics. Finally, we provide a case study in exploratory data analysis with our methods using cooking data, where a recipe's ingredients form a hyperedge and its edge label the cuisine type.

2 PRELIMINARIES AND RELATED WORK

Let $G = (V, E, C, \ell)$ be an edge-labeled (hyper)graph, where V is a set of nodes, E is a set of (hyper)edges, C is a set of categories (or colors), and $\ell : E \to C$ is a function which labels every edge with a category. Often, we just use $C = \{1, 2, \ldots, k\}$, and we can think of ℓ as a coloring of the edges. We use "category", "color", and "label" interchangeably, as these terms appear in different types of literature (e.g., "color" is common for discrete labeling in graph theory and combinatorics). We use $k = |C|$ to denote the number of categories, $E_c \subseteq E$ for the set of edges having label c, and r for the maximum hyperedge size (i.e., *order*), where the size of a hyperedge is the number of nodes it contains (in the case of graphs, $r = 2$).

2.1 Categorical edge clustering objective

Given G, we consider the task of assigning a category (color) to each node in such a way that nodes in category c tend to participate in edges with label c; in this setup, we partition the nodes into k clusters with one category per cluster. We encode the objective function as minimizing the number of "mistakes" in a clustering, where a mistake is an edge that either (i) contains nodes assigned to different clusters or (ii) is placed in a cluster corresponding to a category which is not the same as its label. In other words, the objective is to minimize the number of edges that are not completely contained in the cluster corresponding to the edge's label.

Let Y be a categorical clustering, or equivalently, a coloring of the nodes, where $Y[i]$ denotes the color of node i. Let $m_Y : E \to \{0, 1\}$ be the *category-mistake* function, defined for an edge $e \in E$ by

$$m_Y(e) = \begin{cases} 1 & \text{if } Y[i] \neq \ell(e) \text{ for any node } i \in e, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Then, the *Categorical Edge Label Clustering* objective score for the clustering Y is simply the number of mistakes:

$$\textbf{CatEdgeClus}(Y) = \textstyle\sum_{e \in E} m_Y(e). \tag{2}$$

This form applies equally to hypergraphs; a mistake is a hyperedge with a node placed in a category different from the edge's label.

Our objective can easily be modified for weighted (hyper)graphs. If a hyperedge e has weight w_e, then the category mistake function simply becomes $m_Y(e) = w_e$ if $Y[i] \neq \ell(e)$ for any node i in e and is 0 otherwise. Our results easily generalize to this setting, but we present results in the unweighted case for ease of notation.

2.2 Relation to Correlation Clustering

Our objective function is related to chromatic correlation clustering [12], in which one clusters an edge-colored graph into any number of clusters, and a penalty is incurred for any one of three types of *mistakes*: (i) an edge of color c is placed in a cluster of a different color; (ii) an edge of any color has nodes of two different colors; or (iii) a pair of nodes *not* connected by an edge is placed inside a cluster. This objective is a strict generalization of the classical correlation clustering objective [8].

Our Categorical Edge Clustering objective is similar, except we remove the penalty for placing non-adjacent nodes in the same cluster (mistakes of type (iii)). The chromatic correlation clustering objective treats the absence of an edge between nodes i and j as a strong indication that these nodes should not share the same label. We instead interpret a non-edge simply as missing information: the absence of an edge may be an indication that i and j do not belong together, but it may also be the case that they have a relationship that simply has not been measured. This is a natural assumption with large, sparse real-world graphs, where we rarely have information on all pairs of entities. Another key difference between chromatic correlation clustering and our objective is that in the former, one may form several clusters for the same color. For our objective, merging two separate clusters for the same color can only improve the objective.

Our formulation also leads to several differences in computational tractability. Chromatic correlation clustering is NP-hard in general, and there are several approximation algorithms [6, 12, 13].

The tightest of these is a 4-approximation, though the algorithm is mostly of theoretical interest, as it involves solving an incredibly large linear program. Moreover, the higher-order generalization of simple correlation clustering (without colors) to hypergraphs is more complicated to solve and approximate than standard correlation clustering [27, 31, 43, 45]. We will show that our Categorical Edge Clustering objective can be solved in polynomial time for graphs and hypergraphs with two categories. The problem becomes NP-hard for more than two categories, but we are able to obtain practical 2-approximation algorithms for both graphs and hypergraphs. Our approaches are based on linear programming relaxations that are small enough to be solved quickly in practice.

2.3 Additional related work

There are several methods for clustering general data points that have categorical features [14, 28, 30], but these methods are not designed for clustering graph data. There are also methods for clustering in graphs with attributes [4, 15, 62, 67]; these focus on vertex features and do not connect categorical features to cluster indicators. Finally, there are several clustering approaches for multilayer networks modeling edge types [23, 38, 47], but the edge types are not meant to be indicative of a cluster type.

3 THE CASE OF TWO CATEGORIES

In this section we design algorithms to solve the Categorical Edge Clustering problem when there are only two categories. In this case, both the graph and hypergraph problem can be reduced to a minimum s-t cut problem, which can be efficiently solved.

3.1 An algorithm for graphs

To solve the two-category problem on graphs, we first convert it to an instance of a weighted minimum s-t cut problem on a graph with no edge labels. Recall that E_c is the set of edges with category label c. Given the edge-labeled graph $G = (V, E, C, \ell)$, we construct a new graph $G' = (V', E')$ as follows:

- Introduce a terminal node v_c for each of the two labels $c \in L$, so that $V' = V \cup V_t$ where $V_t = \{v_c \mid c \in L\}$.
- For each label c and each $(i, j) \in E_c$, introduce edges (i, j), (v_c, i) and (v_c, j), all of which have weight $\frac{1}{2}$.

Since there are only two categories c_1 and c_2, let $s = v_{c_1}$ be treated as a source node and $t = v_{c_2}$ be treated as a sink node. The minimum s-t cut problem in G' is defined by

$$\underset{S \subseteq V}{\text{minimize}} \ \ \mathbf{cut}(S \cup s), \tag{3}$$

where $\mathbf{cut}(T)$ is the weight of edges crossing from nodes in $T \subset V'$ to its complement set $\bar{T} = V' \backslash T$. This classical problem that can be efficiently solved in polynomial time, and we have an equivalence with the original two-category edge clustering objective.

Proposition 3.1. *For any $S \subseteq V$, the value of $\mathbf{cut}(S \cup s)$ in G' is equal to the value of $\mathbf{CatEdgeClus}(\{S, \bar{S}\})$, where S and $\bar{S}$ are the clusters for categories c_1 and c_2.*

Proof. Let edge $e = (i, j)$ be a "mistake" in the clustering ($m_Y(e) = 1$) and without loss of generality have color c_1. If i and j are assigned to c_2, then the half-weight edges (i, v_{c_1}) and (j, v_{c_1}) are cut.

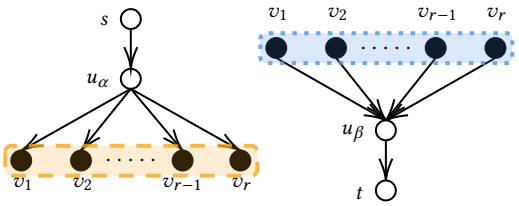


Figure 1: Subgraphs used for the s-t cut reduction of two-color Categorical Edge Clustering in hypergraphs. Here, α and β are hyperedges in the original hypergraph with colors c_1 (orange, left) and c_2 (blue, right).

Otherwise, exactly one of i and j is assigned to c_2. Without loss of generality, let it be i. Then (i, v_{c_1}) and (i, j) are cut. □

Thus, a minimizer for the s-t cut in G' directly gives us a minimizer for our Categorical Edge Clustering objective. We next provide a similar reduction for the case of hypergraphs.

3.2 An algorithm for hypergraphs

We now develop a method to exactly solve our objective in the two-color case with arbitrary order-r hypergraphs, and we again proceed by reducing to an s-t cut problem. Our approach is to construct a subgraph for every hyperedge and paste these subgraphs together to create a new graph $G' = (V', E')$, where minimum s-t cuts produce partitions that minimize the Categorical Edge Clustering objective. A similar construction has been used for a $\mathcal{P}^r$ Potts model in computer vision [36], and our reduction is the first direct application of this approach to network analysis.

We start by adding terminal nodes $s = v_{c_1}$ and $t = v_{c_2}$ (corresponding to categories c_1 and c_2) as well as all nodes in V to V'. For each hyperedge $e = (v_1, \ldots, v_r)$ of G, we add a node u_e to V' and add the following *directed* edges to E' (see also Figure 1):

- If e has label c_1, add $(s, u_e), (u_e, v_1), \ldots, (u_e, v_r)$ to E'.
- If e has label c_2, add $(u_e, t), (v_1, u_e), \ldots, (v_r, u_e)$ to E'.

Again, the minimum s-t cut on G' produces a partition that also minimizes the categorical edge clustering objective, as shown below.

Theorem 3.2. *Let $\mathcal{S}^*$ be the solution to the minimum cut problem. Then the label assignment Y defined by $Y[i] = c_1$ if $i \in S^*$ and $Y[i] = c_2$ if $i \in \bar{S}^*$ minimizes the Categorical Edge Clustering objective.*

Proof. Consider a hyperedge $e = (v_1, \ldots, v_r)$ with label c_2. We show that $m_Y(e)$ precisely corresponds to an s-t cut on the subgraph of G' induced by e (Figure 1, right). If $Y[v_1] = \ldots = Y[v_r] = c_2$, then $v_1, \ldots, v_r \in \bar{S}^*$ and the cost of the minimum s-t-cut is 0 (via placing s by itself). Now suppose at least one of $Y[v_1], \ldots, Y[v_r]$ equals c_1. Without loss of generality, say that $Y[v_1] = c_1$, so $v_1 \in S^*$. If $u_e \in S^*$, we cut (u_e, t) and none of the edges (v_i, u_e) contribute to the cut. If $u_e \in \bar{S}^*$, we cut (v_1, u_e); and it cannot be the case that (v_i, u_e) is cut for $i \neq 1$ (otherwise, we could have reduced the cost of the minimum cut by placing $u_e \in S^*$).

To summarize, if edge e with label c_2 induces a mistake in the clustering, then the cut contribution is 1; otherwise, it is 0. A symmetric argument holds if e has label c_1, using the graph in Figure 1 (left). By additivity, minimizing the s-t cut in G' minimizes the number of mistakes in the Categorical Edge Clustering objective. □

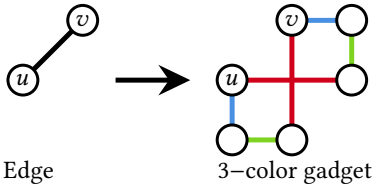


Figure 2: Gadget used for reducing maxcut to 3-color Categorical Edge Clustering. Each gadget has new auxiliary nodes, but u and v may be a part of many 3-color gadgets.

This procedure also works for the special case of graphs. However, G' has more nodes and directed edges in the more general reduction, which can increase running time in practice.

Computational considerations. Both algorithms solve a single minimum cut problem on a graph with $O(T)$ vertices and $O(T)$ edges, where $T = \sum_{e \in E} |e|$ is the sum of hyperedge degrees (this is bounded above by $r|E|$, where r is the order of the hypergraph). In theory, this can be solved in $O(T^2)$ time in the worst case [49]. However, practical performance is often much different than this worst-case running time. That being said, we do find the maximum flow formulations to often be slower than the linear programming relaxations we develop in Section 4. We emphasize that being able to solve the Categorical Edge Clustering objective in polynomial time for two colors is itself interesting, and that the algorithms we use for experiments in Section 5 are able to scale to large hypergraphs.

Considerations for unlabeled edges. Our formulation assumed that all of the (hyper)edges carry a unique label. However, in some datasets, there may be edges with no label or both labels. In these cases, the edge's existence still signals that its constituent nodes should be colored the same — just not with a particular color. A natural augmentation to our objective is then to penalize this edge only when it is not entirely contained in *some* cluster. Our reductions above handle this case by simply connecting the corresponding nodes in V' to both terminals instead of just one.

4 MORE THAN TWO CATEGORIES

We now move to the general formulation of Categorical Edge Clustering when there can be more than two categories or labels. We first show that optimizing the objective in this setting is NP-hard. After, we develop approximation algorithms based on linear programming relaxations and multiway cut problems with theoretical guarantees on solution quality. Many of these algorithms are practical, and we use them in numerical experiments in Section 5.

4.1 NP-hardness of Categorical Edge Clustering

We now prove that the Categorical Edge Clustering objective is NP-hard for the case of three categories. Our proof follows the structure of the NP-hardness reduction for 3-terminal multiway cut [21], and the reduction is from the NP-hard maximum cut (maxcut) problem. Written as a decision problem, this problem seeks to answer if there exists a partition of the nodes of a graph into two sets such that the number of edges cut by the partition is at least K.

Consider an unweighted instance of maxcut on a graph $G = (V, E)$. To convert this into an instance of 3-color Categorical Edge Clustering, we replace each edge $(u, v) \in E$ with the 3-color gadget in Figure 2. We will use the following lemma in our reduction.

Lemma 4.1. *In any node coloring of the 3-color gadget (Figure 2), the minimum number of edges whose color does not match both of its nodes (i.e., number of mistakes in categorical edge clustering) is three. This only occurs when one of $\{u, v\}$ is red and the other is blue.*

Proof. If v is blue and u is red, then we can achieve the minimum three mistakes by clustering each node in the gadget with its horizontal neighbor in Figure 2 or alternatively by placing each node with its vertical neighbor. If u and v are constrained to be in the same cluster, then the optimal solution is to place all nodes in the gadget together, which makes 4 mistakes. It is not hard to check that all other color assignments yield a penalty of 4 or more. □

Now let G' be the instance of 3-color Categorical Edge Clustering obtained by replacing each edge $(u, v) \in E$ with a 3-color gadget.

Theorem 4.2. *There exists a partition of the nodes in G into two sets with K or more cut edges if and only if there is a 3-coloring of the nodes in G' that makes $4|E| - K$ or fewer mistakes.*

Proof. Consider first a cut in $G = (V, E)$ of size $K' \geq K$. Let S_r and S_b denote the two clusters in the corresponding bipartition of G, mapping to red and blue clusters. Consider each $(u, v) \in E$ in turn along with its 3-color gadget. If $(u, v) \in E$ is cut, cluster all nodes in its gadget with their vertical neighbor if $u \in S_b$ and $v \in S_r$, and cluster them with their horizontal neighbor if $u \in S_r$ and $v \in S_b$. Either way, this makes exactly 3 mistakes. If (u, v) is *not* cut, then label all nodes in the gadget red if $u, v \in S_r$, or blue if $u, v \in S_b$, which makes exactly 4 mistakes. The total number of mistakes in G' is then $3K' + 4(|E| - K') = 4|E| - K' \leq 4|E| - K$.

Now start with G' and consider a node coloring that makes $B' \leq B = 4|E| - K$ mistakes. There are $|E|$ total 3-color gadgets in G'. We claim that there must be at least K of these gadgets at which only three mistakes are made. If this were not the case, then assume exactly $H < K$ gadgets where 3 mistakes are made. By Lemma 4.1, there are $|E| - H$ gadgets where at least 4 mistakes are made, so the total number of mistakes is $B' \geq 3H + 4(|E| - H) = 4|E| - H > 4|E| - K$, contradicting our initial assumption. Thus, by Lemma 4.1, there are at least K edges $(u, v) \in E$ where one of $\{u, v\}$ is red and the other is blue, and the maximum cut in G is at least K. □

Consequently, if we can minimize Categorical Edge Clustering in polynomial time, we can solve the maximum cut decision problem in polynomial time, and Categorical Edge Clustering is thus NP-hard. As a natural next step, we turn to approximation algorithms.

4.2 Algorithms based on LP relaxations

We now develop approximation algorithms by relaxing an integer linear programming (ILP) formulation of our problem. We design the algorithms for hypergraphs, with graphs as a special case. Suppose we have an edge-labeled hypergraph $G = (V, E, C, \ell)$ with $C = \{1, \ldots, k\}$, where $E_c = \{e \in E \mid \ell[e] = c\}$. The Categorical Edge Clustering objective can be written as the following ILP:

$$\begin{array}{lll} \min & \sum_{c \in C} \sum_{e \in E_c} x_e & \\ \text{s.t.} & \text{for all } v \in V: & \sum_{c=1}^{k} x_v^c = k - 1 \\ & \text{for all } c \in C,\, e \in E_c: & x_v^c \leq x_e \text{ for all } v \in e \\ & x_v^c, x_e \in \{0, 1\} & \text{for all } c \in C,\, v \in V,\, e \in E. \end{array} \tag{4}$$

Algorithm 1: A simple 2-approximation for Categorical Edge Clustering based on an LP relaxation. Algorithm 2 details a more sophisticated rounding scheme.

Input: Labeled hypergraph $G = (V, E, C, \ell)$.
Output: Label $Y[i]$ for each node $i \in V$.
Solve the LP-relaxation of ILP (4).
for $c \in C$ **do**
  $S_c \leftarrow \{v \in V \mid x_v^c < 1/2\}$.
  for $i \in S_c$ **do** assign $Y[i] \leftarrow c$.
end
Assign unlabeled nodes to an arbitrary $c \in C$.

In this ILP, $x_v^c = 1$ if node v is *not* assigned to category c, and is zero otherwise. The first constraint in (4) ensures that $x_v^c = 0$ for exactly one category. The second constraint says that in any minimizer, $x_e = 0$ if and only if all nodes in e are colored the same as e; otherwise, $x_e = 1$. If we relax the binary constraints in (4):

$$0 \le x_v^c \le 1, \quad 0 \le x_e \le 1,$$

then the ILP is just a linear program (LP) that can be solved in polynomial time.

When $k = 2$, the constraint matrix of the LP relaxation is totally unimodular as it corresponds to the incidence matrix of a balanced signed graph [64]. Thus, all basic feasible solutions for the LP satisfy the binary constraints of the original ILP (4), which is another proof that the two-category problem can be solved in polynomial time.

With more than two categories, the LP solution can be fractional, and we cannot directly determine a node assignment from the LP solution. Nevertheless, solving the LP provides a lower bound on the optimal solution, and we show how to round the result to produce a clustering within a bounded factor of the lower bound. Algorithm 1 contains our rounding scheme, and the following theorem shows that it provides a clustering within a factor of 2 from optimal.

Theorem 4.3. *Algorithm 1 returns at worst a 2-approximation to the Categorical Edge Clustering objective.*

Proof. First, for any $v \in V$, $x_v^c < 1/2$ for at most one category $c \in C$ in the solution. If this were not the case, there would exist two colors a and b such that $x_v^a < 1/2$ and $x_v^b < 1/2$ and

$$\textstyle\sum_{c=1}^{k} x_v^c = x_v^a + x_v^b + \sum_{c' \in C \setminus \{a,b\}} x_v^{c'} < 1 + k - 2 = k - 1,$$

which violates the first constraint of the LP relaxation. Therefore, each node will be assigned to at most one category. Consider any $e \in E_c$ for which all nodes are not assigned to c. This means that there exists at least one node $v \in e$ such that $x_v^c \ge 1/2$. Thus, the Algorithm incurs a penalty of one for this edge, but the LP relaxation pays a penalty of $x_e \ge x_v^c \ge 1/2$. Therefore, every edge mistake will be accounted for within a factor of 2. □

We can get better approximations in expectation with a more sophisticated randomized rounding algorithm (Algorithm 2). In this approach, we form sets S_c^t based on a threshold parameter t so that each node may be included in more than one set. To produce a valid clustering, we first generate a random permutation of colors to indicate an (arbitrary) priority of one color over another. For any $v \in V$ contained in more than one set S_c^t, we assign v to the cluster with highest priority. By carefully setting the parameter t, this approach has better guarantees than Algorithm 1.

Algorithm 2: LP relaxation for Categorical Edge Clustering with a randomized rounding scheme. Theorem 4.4 gives approximation guarantees based on t.

Input: Labeled hypergraph $G = (V, E, C = \{1, 2, \ldots, k\}, \ell)$; rounding parameter $t \in [1/2, 2/3]$.
Output: Label $Y[i]$ for each node $i \in V$.
Solve the LP-relaxation of ILP (4).
$\pi \leftarrow$ uniform random permutation of $\{1, 2, \ldots, k\}$.
for $c = \pi_1, \ldots, \pi_k$ **do**
  $S_c \leftarrow \{v \in V \mid x_v^c < t\}$.
  for $i \in S_c$ **do** $Y[i] \leftarrow \pi(c)$.
end
Assign unlabeled nodes to an arbitrary $c \in C$.

Theorem 4.4. *If $t = k/(2k-1)$, Algorithm 2 returns an at worst $(2 - 1/k)$-approximation for Categorical Edge Clustering in expectation. And if $t = (r+1)/(2r+1)$, Algorithm 2 returns an at worst $(2 - 1/(1+r))$-approximation in expectation.*

Proof. For the choices of t listed in the statement of the theorem, $t \in [1/2, 2/3]$ as long as $r \ge 2$ and $k \ge 2$, which is always true. We say that color c *wants* node v if $v \in S_c$, but this does not automatically mean that v will be colored as c. For any $v \in V$, there exist at most two colors that want v. If v were wanted by more than two colors, this would mean $v \in S_a \cap S_b \cap S_c$ for three distinct colors a, b, c. This leads to a violation of the first constraint in (4):

$$x_v^a + x_v^b + x_v^c + \sum_{i : i \notin \{a,b,c\}} x_v^i < 3t + (k-3) \le 2 + (k-3) = (k-1).$$

Consider an arbitrary $t \in (1/2, 2/3)$. We can bound the expected number of mistakes made by Algorithm 2 and pay for them individually in terms of the LP lower bound. To do this, we consider a single hyperedge $e \in E_c$ with color c and bound the probability of making a mistake and the LP cost of this hyperedge.

Case 1: $x_e \ge t$. In this case, we are guaranteed to make a mistake at edge e, since $x_e \ge t$ implies there is some node $v \in e$ such that $x_v^c \ge t$, and so $v \notin S_c$. However, because the LP value at this edge is $x_e \ge t$, we pay for our mistake within a factor $1/t$.

Case 2: $x_e < t$. Now, color c wants every node in the hyperedge $e \in E_c$. If no other colors want any node $v \in e$, then Algorithm 2 will not make a mistake at e, and we have no mistake to account for. Assume then that there is some node $v \in e$ and a color $c' \ne c$ such that c' wants v. This implies that $x_v^{c'} < t$, from which we have that $x_v^c \ge 1 - x_v^{c'} > 1 - t$ (to satisfy the first inequality in (4)). Thus,

$$x_e \ge x_v^{c'} > 1 - t. \tag{5}$$

This gives a lower bound of $1 - t$ on the contribution of the LP objective at edge e.

In the worst case, each $v \in e$ may be wanted by a *different* $c' \ne c$, and the number of colors other than c that want some node in e is bounded above by $B_1 = k - 1$ and $B_2 = r$. We avoid a mistake at e if and only if c has higher priority than all of the alternative colors,

where priority is established by the random permutation π. Thus,

$$\mathbf{Pr}[\text{mistake at e} \mid x_e < t] \leq \frac{B_i}{B_i+1} = \min\left\{\frac{r}{r+1}, \frac{k-1}{k}\right\}. \quad (6)$$

Recall from (5) that the LP pays $x_e > 1-t$. Therefore, the expected cost at a hyperedge $e \in E_c$ satisfying $x_e < t$ is at most $\frac{B_i}{(1-t)(B_i+1)}$ in expectation. Taking the worst approximation factor from Case 1 and Case 2, Algorithm 2 will in expectation provide an approximation factor of $\max\left\{\frac{1}{t}, \frac{B_i}{(1-t)(B_i+1)}\right\}$. This will be minimized when the approximation bounds from Cases 1 and 2 are equal, which occurs when $t = \frac{B_i+1}{2B_i+1}$. If $B_i = k-1$, then $t = \frac{k-1}{2k-1}$ and the expected approximation factor is $2 - 1/k$. And if $B_i = r$, then $t = \frac{r}{2r+1}$ and the expected approximation factor is $2 - 1/(r+1)$. □

For the graph case ($r = 2$), this theorem implies a $\frac{5}{3}$-approximation for Categorical Edge Clustering with any number of categories.

Computational considerations. The linear program has $O(|E|)$ variables and sparse constraints, which written as a matrix inequality would have $O(T)$ non-zeros, where T is again the sum of hyperedge degrees. Improving the best theoretical running times for solving linear programs is an active area of research [19, 40], but practical performance of solving linear programs is often much different than worst-case guarantees. In Section 5, we show that a high-performance LP solver from Gurobi is extremely efficient in practice, finding solutions in seconds on hypergraphs with several categories and tens of thousands of hyperedges in tens of seconds.

4.3 Algorithms based on multiway cut

We now provide alternative approximations based on multiway cut, similar to the reductions from Section 3. Again, we develop this technique for general hypergraphs and graphs are a special case.

Suppose we have an edge-labeled hypergraph $G = (V, E, C, \ell)$. We construct a new graph $G' = (V', E')$ as follows. First, introduce a terminal node v_c for each category $c \in C$, so that $V' = V \cup \{v_c \mid c \in C\}$. Second, for each hyperedge $e = \{v_1, \ldots, v_r\} \in E$, add a clique on nodes $v_1, \ldots, v_r, v_{\ell[e]}$ to E', where each edge in the clique has weight $1/r$. (Overlapping cliques are just additive on the weights.)

The multiway cut objective is the number of cut edges in any partition of the nodes into k clusters such that each cluster contains exactly one of the terminal nodes. We can associate each cluster with a category, and any clustering Y of nodes in Categorical Edge Clustering for G can be mapped to a candidate partition for multiway cut in G'. Let $\mathbf{MultiwayCut}(Y)$ denote the value of the multiway cut objective for the clustering Y. The next result relates multiway cut to Categorical Edge Clustering.

Theorem 4.5. *For any clustering Y,*

$$\mathbf{CatEdgeClus}(Y) \leq \mathbf{MultiwayCut}(Y) \leq \frac{r+1}{2}\mathbf{CatEdgeClus}(Y).$$

Proof. Let $e = \{v_1, \ldots, v_r\}$ with label $c = \ell[e]$ be a hyperedge in G. We can show that the bounds hold when considering the associated clique in G' and then apply additivity. First, if e is not a mistake in the Categorical Edge Clustering, then no edges are cut in the clique. If e is a mistake in the Categorical Edge Clustering, then there are some edges cut in the associated clique. The smallest possible contribution to the multiway cut objective occurs when all but one node is assigned to c. Without loss of generality, consider this to be v_1, which is in r cut edges: $(r-1)$ corresponding to the edges from v_1 to other nodes in the hyperedge, plus one for the edge from v_1 to the terminal v_c. Each of the r cut edges has weight $1/r$, so the multiway cut contribution is 1.

The largest possible cut occurs when all nodes in e are colored differently from e. In this case, the edges incident to each node in the clique are all cut. For any one of these nodes, the sum of edge weights incident to that node equals 1 by the same arguments as above. This cost is incurred for each of the r nodes in the hyperedge plus the terminal node v_c, for a total weight of $r+1$. Since each edge is counted twice, the actual penalty is $(r+1)/2$. □

Computational considerations. Minimizing the multiway cut objective is NP-hard [21], but there are many approximation algorithms. Theorem 4.5 implies that any p-approximation for multiway cut provides a $p(r+1)/2$-approximation for Categorical Edge Clustering. For example, the simple isolating cuts heuristic yields a $\frac{r+1}{2}(2 - \frac{2}{k})$-approximation, and more sophisticated algorithms provide a $\frac{r+1}{2}(\frac{3}{2} - \frac{1}{k})$-approximation [18]. For our experiments, we use the isolating cut approach, which solves $O(k)$ maximum flow problems on a graph with $O(r|E|)$ vertices and $O(r^2|E|)$ edges. This can be expensive in practice. We will find that the LP relaxation performs better in terms of solution quality and running time.

A node-weighted multiway cut reduction. We also provide an approximation based on a *direct* reduction to a node-weighted multiway cut (NWMC) problem that is of theoretical interest. As above, suppose we have an edge-labeled hypergraph $G = (V, E, C, \ell)$. We construct a new graph $G' = (V', E')$ as follows. First, introduce a terminal node v_c for each category $c \in C$, so that $V' = V \cup \{v_c \mid c \in C\}$. Assign infinite weights to all nodes in V'. Next, for each hyperedge $e = \{v_1, \ldots, v_r\} \in E$, add an auxiliary node v_e with weight 1. Next, append edges $(v_e, v_1), \ldots, (v_e, v_r)$ as well as (v_c, v_e) for $\ell(e) = c$ to E'. It straightforward to check that deleting v_e corresponds to making a mistake at hyperedge e. Thus an optimizer of NWMC on G' is also an optimizer of Categorical Edge Clustering on G.

Solving NWMC is also NP-hard [29], and there are again well-known approximation algorithms. The above discussion implies any p-approximation to NWMC also provides a p-approximation for Categorical Edge Clustering. For example, an LP-based algorithm has a $2(1-1/k)$-approximation [29]. This approximation is better but the LPs are too large to be practical; however, the improvement of a direct algorithm suggests room for better theoretical results.

4.4 Approximation through a linear objective

The Categorical Edge Clustering objective assigns a penalty of 1 regardless of the *proportion* of the nodes in a hyperedge which are clustered away from hyperedge's color. Although useful, we might consider alternative penalties that value the *extent* to which each hyperedge is satisfied in the final clustering. One natural penalty for a hyperedge of color c is the number of nodes within that hyperedge that are not clustered into that color. With such a "linear" mistake function, we define the Categorical Node Clustering Objective as

$$CatNodeClus(Y) = \textstyle\sum_{e \in E} m'_Y(e), \text{ where } m'_Y(e) = \sum_{i \in e} I_{Y[i] \neq \ell(e)}.$$

It turns out that this objective is optimized with a simple majority vote algorithm that assigns a node to the majority color of all hyperedges that conatin it.

Theorem 4.6. *The majority vote algorithm yields an optimizer of the Categorical Node Clustering (linear) objective.*

Proof. Suppose node u is contained in J_i hyperedges of color i. Without loss of generality, assume $J_1 \geq \ldots \geq J_k$. The cost of assigning u to c is $C_c = \sum_{j \neq c} J_j$, which is minimized for $c = 1$. □

In Section 5, we will see that the majority vote solution provides a good approximation to the optimizer of the Categorical Edge Clustering objective. The reason is that the cost of a hyperedge under the linear objective is at most r while that cost under the Categorical Edge Clustering objective is just 1, which makes majority vote an r-approximation algorithm.

Theorem 4.7. *The majority vote algorithm provides an r-approximation for Categorical Edge Clustering.*

5 EXPERIMENTS

We now run four types of numerical experiments to demonstrate our methodology. First, we show that our algorithms indeed work well on a broad range of datasets at optimizing our objective function and discover that our LP relaxation tends be extremely effective in practice, often finding an optimal solution (i.e., matching the lower bound). After, we show that our approach is superior to competing baselines in categorical community detection experiments where edges are colored to signal same-community membership. Next, we show how to use timestamped edge information as a categorical edge label, and demonstrate that our method can find clusters that preserve temporal information better than methods that only look at graph topology, without sacrificing performance on topological metrics. Finally, we present a case study on a network of cooking ingredients and recipes to show that our methods can also be used for exploratory data analysis. Our code and datasets are available at https://github.com/nveldt/CategoricalEdgeClustering.

5.1 Analysis on Real Graphs and Hypergraphs

We first evaluate our methods on several real-world edge-labeled graphs and hypergraphs in terms of Categorical Edge Clustering. The purpose of these experiments is to show that our methods can optimize the objective quickly and accurately and to demonstrate that our methods find global categorical clustering structure better than natural baseline algorithms. All experiments ran on a laptop with a 2.2 GHz Intel Core i7 processor and 8 GB of RAM. We implemented our algorithms in Julia, using Gurobi software to solve the linear programs.

Datasets. Table 1 provides summary statistics of the datasets we use, and we briefly describe them. *Brain* [20] is a graph where nodes represent brain regions from an MRI. There are two edge categories: one for connecting regions with high fMRI correlation and one for connecting regions with similar activation patterns. In the Drug Abuse Warning Network (*DAWN*) [58], nodes are drugs, hyperedges are combinations of drugs taken by a patient prior to an emergency room visit, and edge categories indicate the patient disposition (e.g., "sent home" or "surgery"). The *MAG-10* network is a subset of the Microsoft Academic Graph [57] where nodes are authors, hyperedges correspond to a publication from those authors, and there are 10 edge categories which denote the computer science conference publication venue (e.g., "WWW" or "KDD"). If the same set of authors published at more than one conference, we used the most common venue as the category, discarding cases where there is a tie. In the *Cooking* dataset [33], nodes are food ingredients, hyperedges are recipes made from combining multiple ingredients, and categories indicate cuisine (e.g., "Southern-US" or "Indian"). Finally, the *Walmart-Trips* dataset is made up of products (nodes), groups of products purchased in a single shopping trip (hyperedges), and categories are 44 unique "trip types" classified by Walmart [32].

Algorithms. We use two algorithms that we developed in Section 4. The first is the simple 2-approximation rounding scheme outlined in Algorithm 1, which we refer to as *LP-round* (*LP*) (in practice, this performs as well as the more sophisticated algorithm in Algorithm 2 and has the added benefit of being deterministic). The second is *Cat-IsoCut* (*IC*), which runs the standard isolating cut heuristic [21] on an instance of multiway cut derived from the Categorical Edge Clustering problem, as outlined in Section 4.3.

The first baseline we compare against is *Majority Vote* (*MV*) discussed in Section 4.4: node i is assigned to category c if c is the most common edge type in which i participates. The *MV* result is also the default cluster assignment for *IC*, since in practice this method leaves some nodes unattached from all terminal nodes.

The other baselines are *Chromatic Balls* (*CB*) and *Lazy Chromatic Balls* (*LCB*) — two algorithms for chromatic correlation clustering [12]. These methods repeatedly select an unclustered edge and greedily grow a cluster around it by adding nodes that share edges with the same label. Unlike our methods, *CB* and *LCB* distinguish between category (color) assignment and cluster assignment: two nodes may be colored the same but placed in different clusters. To provide a uniform comparison among methods, we merge distinct clusters of the same category into one larger cluster. These methods are *not* designed for hypergraph clustering, but we still use them for comparison by reducing a hypergraph to an edge-labeled graph, where nodes i and j share an edge in category c if they appear together in more hyperedges of category c than any other.

Results. Table 1 reports how well each algorithm solves the Categorical Edge Clustering objective. We report the approximation guarantee (the ratio between each algorithm's output and the LP lower bound), as well as the *edge satisfaction*, which is the fraction of hyperedges that end up inside a cluster with the correct label. Maximizing edge satisfaction is equivalent to minimizing the number of edge label mistakes but provides an intuitive way to interpret and analyze our results. High edge satisfaction scores imply that a dataset is indeed characterized by large groups of objects that tend to interact in a certain way with other members of the same group. A low satisfaction score indicates that a single label for each node may be insufficient to capture the intricacies of the data.

In all cases, the LP solution is integral or nearly integral, indicating that *LP* does an extremely good job solving the original NP-hard objective, often finding an exactly-optimal solution. As a result, it outperforms all other methods on all datasets. Furthermore, on nearly all datasets, we can solve the LP within a few seconds or a few minutes. *Walmart* is the exception–given the large number of categories, the LP contains nearly 4 million variables, and far more constraints. Other baseline algorithms can be faster, but they do not perform as well in solving the objective.

Table 1: Summary statistics of datasets — number of nodes $|V|$, number of (hyper)edges $|E|$, maximum hyperedge size r, and number of categories k — along with Categorical Edge Clustering performance for the algorithms *LP-round* (LP), *Majority Vote* (MV), *Cat-IsoCut* (IC), *ChromaticBalls* (CB) and *LazyChromaticBalls* (LCB). Performance is listed in terms of the approximation guarantee given by the LP lower bound (lower is better) and in terms of the edge satisfaction, which is the fraction of edges that are *not mistakes* (higher is better; see Eq. (2)). Our LP method performs the best overall and can even find exactly (or nearly) optimal solutions to the NP-hard objective by matching the lower bound. We also report the running times for rough comparison, though our implementations are not optimized for efficiency. Due to its simplicity, MV is extremely fast.

					Approx. Guarantee					Edge Satisfaction					Runtime (in seconds)				
Dataset	$\|V\|$	$\|E\|$	r	k	LP	MV	IC	CB	LCB	LP	MV	IC	CB	LCB	LP	MV	IC	CB	LCB
Brain	638	21180	2	2	1.0	1.01	1.27	1.56	1.41	0.64	0.64	0.55	0.44	0.5	1.8	0.0	1.9	0.4	0.8
MAG-10	80198	51889	25	10	1.0	1.18	1.37	1.44	1.35	0.62	0.55	0.48	0.45	0.49	51	0.1	203	333	699
Cooking	6714	39774	65	20	1.0	1.21	1.21	1.23	1.24	0.2	0.03	0.03	0.01	0.01	72	0.0	1223	4.6	6.7
DAWN	2109	87104	22	10	1.0	1.09	1.0	1.31	1.15	0.53	0.48	0.53	0.38	0.46	13	0.0	190	0.3	0.4
Walmart-Trips	88837	65898	25	44	1.0	1.2	1.19	1.26	1.26	0.24	0.09	0.09	0.04	0.05	7686	0.2	68801	493	1503

The high edge satisfaction scores indicate that our method does the best job identifying sets of nodes which *as a group* tend to participate in one specific type of interaction. In contrast, the *MV* algorithm identifies nodes that individually exhibit a certain behavior, but the method does not necessarily form clusters of nodes that as a group interact in a similar way. Because our *LP* method outperforms our *IC* approach on all datasets in terms of both speed and accuracy, in the remaining experiments we focus only on comparing *LP* against other competing algorithms.

5.2 Categorical Edge Community Detection

Next we demonstrate the superiority of *LP* in detecting communities of nodes with the same node labels (i.e., *categorical communities*), based on labeled edges between nodes. We perform experiments on synthetic edge-labeled graphs, as well as two real-world datasets, where we reveal edge labels indicative of the ground truth node labels and see how well we can recover the node labels.

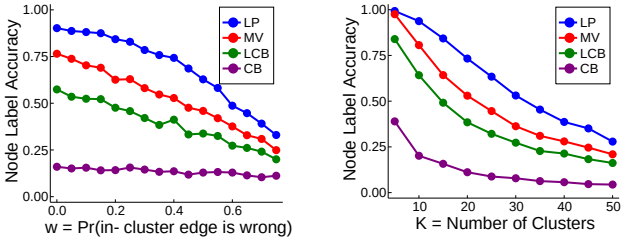
Synthetic Model. We use the synthetic random graph model of Bonchi et al. for chromatic correlation clustering [12]. A user specifies the number of nodes n, colors L, and clusters K, as well as edge parameters p, q, and w. The model first assigns nodes to clusters uniformly at random, and then assigns clusters to colors uniformly at random. (Due to the random assignment, some clusters and colors may not be sampled. Thus, K and L are upper bounds on the number of distinct clusters and unique colors.) For nodes i and j in the same cluster, the model connects them with an edge with probability p. With probability $1 - w$, the edge is the same color as i and j. Otherwise, it is a uniform random color. If i and j are in different clusters, an edge is drawn with probability q and given a uniform random color. We will also use a generalization of this model to synthetic r-uniform hypergraphs. The difference is that we assign colored hyperedges to r-tuples of the n nodes, rather than just pairs, and we assign each cluster to a unique color.

Synthetic Graph Results. We set up two experiments, where performance is measured by the fraction of nodes placed in the correct cluster (node label accuracy). In the first, we form graphs with $n = 1000$, $p = 0.05$, and $q = 0.01$, fixing $L = K = 15$ (which in practice leads to graphs with 15 clusters and typically between 8 and 12 distinct edge and cluster colors). We then vary the noise parameter w from 0 to 0.75 in increments of 0.05. Figure 3a reports the median accuracy over 5 trials of each method for each value of w. In the second, we fix $w = 0.2$, and vary the number of clusters K from 5 to 50 in increments of 5 with $L = K$. Figure 3b reports the median accuracy over 5 trials for each value of K.

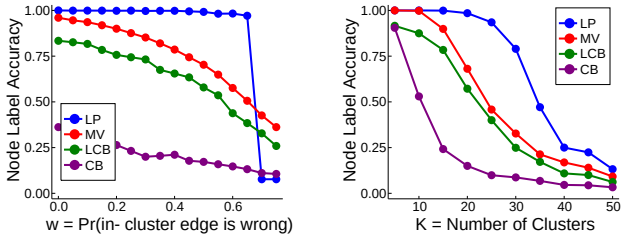
In both experiments, our *LP* method substantially outperforms the others, although the methods perform more similarly for high noise levels or large numbers of clusters. In both experiments, we found that *LP* similarly outperformed other methods in terms of Adjusted Rand Index and F1-score, followed in performance by MV. Cluster identification scores for LCB and CB were particularly low (ARI scores always below 0.02), as these methods formed far too many clusters. Given that LCB and CB are specifically designed for settings where $K \gg L$, we ran another experiment with $L = 20$ and K ranging from 50 to 200. Even in this setting, *LP* achieved higher accuracy, as well as better ARI scores when $K \leq 100$. LCB and CB only obtained better ARI scores in parameter regimes where all algorithms had ARI scores below 0.1. We include more details in an extended online version of our paper.

Synthetic Hypergraph Results. We ran similar experiments on synthetic 3-uniform hypergraphs. We again set $n = 1000$ and used $p = 0.005$ and $q = 0.0001$ for intra-cluster and inter-cluster hyperedge probabilities. In one experiment, we fixed $L = 15$ and varied w, and in another we fixed $w = 0.2$ and varied the number of clusters L. Figures 3c and 3d shows the accuracies. Again, *LP* tends to have the best performance. When $L = 15$, our method achieves nearly perfect accuracy for $w \leq 0.6$. However, we observe performance sensitivity when the noise is too large: when w increases from 0.6 to 0.65, the output of *LP* no longer tracks the ground truth cluster assignment. This occurs despite the fact that the LP solution is integral, and we are in fact optimally solving the Categorical Edge Clustering objective. We conjecture this sharp change in accuracy is due to an information theoretic detectability threshold, which depends on parameters of the synthetic model.

Academic Department Labels in an Email Network. To test the algorithms on real-world data, we use the *Email-Eu-core* network [41, 63]. Nodes in the graph represent researchers at a European institution, edges indicate email correspondence (we consider the edges as undirected), and nodes are labeled by the departmental affiliation of each researcher. We wish to test how well each method can identify node labels, if we assume we have access to a (perhaps

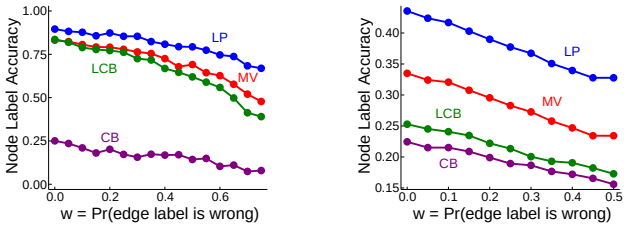


(a) Graphs: Varying Noise w (b) Graphs: Varying # Clusters



(c) Hypergraphs: Varying Noise w (d) Hypergraphs: Varying # Clusters

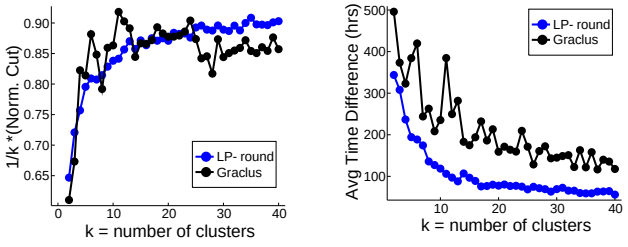
Figure 3: (a)–(b): Performance of algorithms on a synthetic graph model for chromatic correlation clustering [12]. Across a range of parameters, our *LP* method outperforms competing methods in predicting the ground truth label of the nodes. (c)–(d): In experiments on synthetic 3-uniform hypergraphs, *LP* performs well for most parameter regimes but there is some sensitivity to the very noisy setting.



(a) Email-Eu-core (b) Walmart-Products

Figure 4: Accuracy in clustering nodes in real-world datasets when edge labels are a noisy signal for ground truth node cluster membership. For both an email graph (a) and a product co-purchasing hypergraph (b), our *LP-Round* method consistently outperforms other methods.

noisy and imperfect) mechanism for associating emails with labels for inter-department and intra-department communication. To model such a mechanism, we generate edge categories in a manner similar to the synthetic above. An edge inside of a cluster (i.e., an email within the same department) is given the correct department label with probability $1 - w$, and a random label with probability w. An edge between two members of different departments is given a uniform random label. Figure 4a reports each algorithm's ability to detect department labels when w varies from 0 to 0.75. Our *LP* method returns the best results in all cases, and is robust in detecting department labels even in the high-noise regime.



(a) Normalized Cut (b) Inner edge time difference

Figure 5: Results for *LP* and *Graclus* in clustering a temporal network. Our *LP* method is competitive for *Graclus*'s objective (normalized cut; left), while preserving the temporal structure of network much better (right).

Product Categories. The *Walmart-Trips* dataset from Section 5.1 also has product information. We assigned products to one of ten broad departments in which they appear on walmart.com (e.g., "Clothing, Shoes, and Accessories") to construct a *Walmart-Products* hypergraph with ground truth node labels. Recall that hyperedges are sets of co-purchased products. We generate noisy hyperedge labels as before, with $1 - w$ as the probability that a hyperedge with nodes from a single department will have the correct label. Results are reported in Figure 4b, and our LP-round method can detect true departments at a much higher rate than the other methods.

5.3 Temporal Community Detection

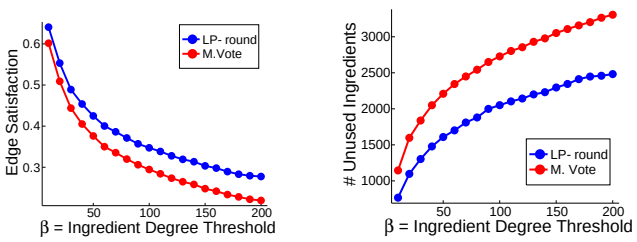
In the next experiment, we show how our framework can be used to identify communities of nodes in a temporal network, where we use timestamps on edges as a type of categorical label that two nodes should be clustered together. For data, we use the *CollegeMsg* network [51], which records private messages (time-stamped edges) between 1899 users (nodes) of a social media platform at UC-Irvine.

Removing timestamps and applying a standard graph clustering algorithm would be a standard approach to identify communities of users. However, this loses the explicit relationship with time. As an alternative, we convert timestamps into discrete edge labels by ordering edges with respect to time and separating them into k equal-sized bins representing time windows. Optimizing Categorical Edge Clustering then corresponds to clustering users into time windows, in order to maximize the number of private messages that occur between users in the same time window. In this way, our framework can identify *temporal communities* in a social network, i.e., groups of users that are highly active in sending each other messages *within a short period of time*.

We construct edge-labeled graphs for different values of k, and compare *LP* against clusterings obtained by discarding time stamps and running *Graclus* [22], a standard graph clustering algorithm. *Graclus* seeks to cluster the nodes into k disjoint clusters $S_1, \ldots, S_k$ to minimize the normalized cut objective:

$$\mathbf{Ncut}(S_1, S_2, \ldots, S_k) = \sum_{i=1}^{k} \frac{\mathbf{cut}(S_i)}{\mathbf{vol}(S_i)},$$

where $\mathbf{cut}(S)$ is the number of edges leaving S, and $\mathbf{vol}(S)$ is the *volume* of S, i.e., the number of edge end points in S. Figure 5a shows that *LP* is in fact competitive with *Graclus* in finding clusterings with small normalized cut scores, even though *LP* is designed for a



(a) Edge Satisfaction (b) Unused ingredients

Figure 6: As β increases, we discard fewer high-degree ingredients before clustering the rest. Our method always "makes" more recipes (higher edge satisfaction) and "wastes" fewer ingredients (smaller number of unused ingredients).

different objective. However, *LP* still avoids cutting edges, and it finds clusterings that also have small normalized cut values. The other goal of *LP* is to place few edges in a cluster with the wrong label, which in this scenario corresponds to clustering messages together if they were sent close in time. We therefore also measure the average difference between timestamps of interior edges and the average time stamp in each clustering, i.e.,

$$\textbf{AvgTimeDiff}(S_1, \ldots, S_k) = \frac{1}{|E_{\text{int}}|} \sum_{i=1}^{k} \sum_{e \in E_i} |\text{timestamp}(e) - \mu_i|,$$

where E_{int} is the set of interior edges completely contained in some cluster, E_i is the set of interior edges of cluster S_i, and μ_i is the average time stamp in E_i. Not surprisingly, this value tends to be large for *Graclus*, since this method ignores timestamps. However, Figure 5b shows that this value tends to be small for *LP*, indicting that it is indeed detecting clusters of users that are highly interactive within a specific short period of time.

5.4 Analysis of the Cooking Hypergraph

Finally, we apply our framework and LP-round algorithm to gain insights into the *Cooking* hypergraph dataset from Section 5.1, demonstrating our methodology for exploratory data analysis. An edge in this hypergraph is a set of ingredients for a recipe, and each recipe is categorized according to cuisine. Categorical Edge Clustering thus corresponds to separating ingredients among cuisines, in a way that maximizes the number of recipes whose ingredients are all in the same cluster (see Ahn et al. [3] for related analyses).

Table 1 shows that only 20% of the recipes can be made (i.e., a 0.2 edge satisfaction) after partitioning ingredients among cuisine types. This is due to the large number of common ingredients such as salt and olive oil that are shared across many cuisines (a problem in other recipe network analyses [60]). We negate the negative effect of high-degree nodes as follows. For an ingredient i, let d_i^c be the number of recipes of cuisine c containing i. Let $M_i = \max_c d_i^c$ measure *majority degree* and $T_i = \sum_t d_i^c$ the *total degree*. Note that $B_i = T_i - M_i$ is a lower bound on the number of hyperedge mistakes we will make at edges incident to node i. We can refine the original dataset by removing all nodes with B_i greater than some β.

Making recipes or wasting ingredients. Figure 6a shows edge satisfaction scores for *LP* and *MV* when we cluster for different β. When $\beta = 10$, edge satisfaction is above 0.64 with *LP*. As β increases, edge satisfaction decreases, but *LP* outperforms *MV* in all cases. We also consider a measure of "ingredient waste" for each method. An ingredient is *unused* if we cannot make any recipes by combining the ingredient with other ingredients in its cluster. A low number of unused ingredients indicates that a method forms clusters where ingredients combine together well. Figure 6b shows the number of unused ingredients as β varies. Again, *LP* outperforms *MV*.

Table 2: Examples of ingredients and recipes from special clusters identified by *LP*, but not *Majority Vote*.

French Fruit-Based Desserts ($\beta = 70$) **Ingredients:** ruby red grapefruit, strawberry ice cream, dry hard cider, icing, prunes, tangerine juice, sour cherries. **Recipes:** 1. {almond extract, bittersweet chocolate, sugar, sour cherries, brioche, heavy cream, unsalted butter, kirsch}, 2. {large egg yolks, ruby red grapefruit, dessert wine, sugar}
Brazilian Caipirinha Recipes ($\beta = 170$) **Ingredients:** simple syrup, light rum, ice, superfine sugar, key lime, coco, kumquats, liquor, mango nectar, vanilla essence **Recipes:** {cachaca, ice} + 1. { lime juice, kumquats, sugar}, 2. {lime, fruit puree, simple syrup}, 3. { superfine sugar, lime juice, passion fruit juice}, 4. { sugar, liquor, mango nectar, lime, mango}

Specific ingredient and recipe clusters. We finally highlight specific ingredient clusters that *LP* identifies but *MV* does not. When $\beta = 170$, *LP* places 10 ingredients with the Brazilian cuisine which *MV* does not, leading to 23 extra recipes that are unique to *LP*. Of these, 21 correspond to variants of the Caipirinha, a popular Brazilian cocktail. When $\beta = 70$, 24 ingredients and 24 recipes are unique to the French cuisine cluster of *LP*. Of these, 18 correspond to desserts, and 14 have a significant fruit component. Table 2 has examples of ingredients and recipes from both these clusters.

6 DISCUSSION

We have developed a computational framework for clustering nodes of hypergraphs when edges have categorical labels that signal node similarity. With two categories, our clustering objective can be solved in polynomial time. For general problems, our linear programming relaxations provide 2-approximation or even better guarantees, which are far tighter than what is seen in the related literature on correlation clustering. This method is also extremely effective in practice. Amazingly, our LP-round algorithm often actually minimizes our NP-hard objective (certified through integral solutions) on hypergraphs with tens of thousands of edges in just tens of seconds. The approach also works well in problems when performance is measured in terms of some sort of ground truth labeling, outperforming baselines by a substantial margin.

For the special cases of two-category graphs and rank-3 hypergraphs, the Categorical Edge Clustering objective is a "regular energy function" within the energy minimization framework of computer vision [37]. This provides alternative polynomial time algorithms in these cases. However, energy minimization approaches do not work for two important regimes: more than two categories, or in general hypergraphs (in the latter, the penalties are no longer a semi-metric, which is needed for approximation algorithms [16]).

Acknowledgments. This research was supported by NSF Award DMS-1830274, ARO Award W911NF19-1-0057, and ARO MURI.

REFERENCES

[1] Evrim Acar, Daniel M. Dunlavy, and Tamara G. Kolda. 2009. Link prediction on evolving data using matrix and tensor factorizations. In *2009 IEEE International Conference on Data Mining Workshops*. IEEE, 262–269.

[2] Sameer Agarwal, Kristin Branson, and Serge Belongie. 2006. Higher order learning with graphs. In *Proceedings of the 23rd International Conference on Machine Learning - ICML '06*. ACM Press. https://doi.org/10.1145/1143844.1143847

[3] Yong-Yeol Ahn, Sebastian E. Ahnert, James P. Bagrow, and Albert-László Barabási. 2011. Flavor network and the principles of food pairing. *Scientific Reports* 1, 1 (Dec. 2011). https://doi.org/10.1038/srep00196

[4] Leman Akoglu, Hanghang Tong, Brendan Meeder, and Christos Faloutsos. 2012. PICS: Parameter-free Identification of Cohesive Subgroups in Large Attributed Graphs. In *Proceedings of the 2012 SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9781611972825.38

[5] Réka Albert and Albert-László Barabási. 2002. Statistical mechanics of complex networks. *Reviews of Modern physics* 74, 1 (2002), 47.

[6] Yael Anava, Noa Avigdor-Elgrabli, and Iftah Gamzu. 2015. Improved Theoretical and Practical Guarantees for Chromatic Correlation Clustering. In *Proceedings of the 24th International Conference on World Wide Web (WWW '15)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 55–65. https://doi.org/10.1145/2736277.2741629

[7] Francesca Arrigo, Desmond J Higham, and Francesco Tudisco. 2019. A framework for second order eigenvector centralities and clustering coefficients. *arXiv:1910.12711* (2019).

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

[9] Austin R Benson. 2019. Three hypergraph eigenvector centralities. *SIAM Journal on Mathematics of Data Science* 1, 2 (2019), 293–312.

[10] Austin R. Benson, Rediet Abebe, Michael T. Schaub, Ali Jadbabaie, and Jon Kleinberg. 2018. Simplicial closure and higher-order link prediction. *Proceedings of the National Academy of Sciences* 115, 48 (2018), E11221–E11230.

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

[12] Francesco Bonchi, Aristides Gionis, Francesco Gullo, Charalampos E. Tsourakakis, and Antti Ukkonen. 2015. Chromatic Correlation Clustering. *ACM Trans. Knowl. Discov. Data* 9, 4, Article 34 (June 2015), 24 pages. https://doi.org/10.1145/2728170

[13] Francesco Bonchi, Aristides Gionis, Francesco Gullo, and Antti Ukkonen. 2012. Chromatic Correlation Clustering. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '12)*. ACM, New York, NY, USA, 1321–1329. https://doi.org/10.1145/2339530.2339735

[14] Shyam Boriah, Varun Chandola, and Vipin Kumar. 2008. Similarity measures for categorical data: A comparative evaluation. In *Proceedings of the 2008 SIAM International Conference on Data Mining*. SIAM, 243–254.

[15] Cecile Bothorel, Juan David Cruz, Mateo Magnani, and Barbora Micenková. 2015. Clustering attributed graphs: Models, measures and methods. *Network Science* 3, 3 (March 2015), 408–444. https://doi.org/10.1017/nws.2015.9

[16] Y. Boykov, O. Veksler, and R. Zabih. 2001. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23, 11 (Nov 2001), 1222–1239. https://doi.org/10.1109/34.969114

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

[18] Gruia Calinescu, Howard Karloff, and Yuval Rabani. 2000. An Improved Approximation Algorithm for MULTIWAY CUT. *J. Comput. System Sci.* 60, 3 (2000), 564 – 574. https://doi.org/10.1006/jcss.1999.1687

[19] Michael B Cohen, Yin Tat Lee, and Zhao Song. 2019. Solving linear programs in the current matrix multiplication time. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. ACM, 938–942.

[20] Nicolas A. Crossley, Andrea Mechelli, Petra E. Vértes, Toby T. Winton-Brown, Ameera X. Patel, Cedric E. Ginestet, Philip McGuire, and Edward T. Bullmore. 2013. Cognitive relevance of the community structure of the human brain functional coactivation network. *Proceedings of the National Academy of Sciences* 110, 28 (2013), 11583–11588.

[21] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. 1994. The Complexity of Multiterminal Cuts. *SIAM J. Comput.* 23 (1994), 864–894.

[22] 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. https://doi.org/10.1109/tpami.2007.1115

[23] Manlio De Domenico, Vincenzo Nicosia, Alexandre Arenas, and Vito Latora. 2015. Structural reducibility of multilayer networks. *Nature Communications* 6, 1 (2015). https://doi.org/10.1038/ncomms7864

[24] Yuxiao Dong, Nitesh V. Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable Representation Learning for Heterogeneous Networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM Press. https://doi.org/10.1145/3097983.3098036

[25] David Easley, Jon Kleinberg, et al. 2012. *Networks, Crowds, and Markets*. Cambridge Books.

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

[27] Takuro Fukunaga. 2018. LP-Based Pivoting Algorithm for Higher-Order Correlation Clustering. In *Computing and Combinatorics*, Lusheng Wang and Daming Zhu (Eds.). Springer International Publishing, Cham, 51–62.

[28] Venkatesh Ganti, Johannes Gehrke, and Raghu Ramakrishnan. 1999. CACTUS-clustering categorical data using summaries. In *KDD*, Vol. 99. 73–83.

[29] Naveen Garg, Vijay V Vazirani, and Mihalis Yannakakis. 2004. Multiway cuts in node weighted graphs. *Journal of Algorithms* 50, 1 (2004), 49–61.

[30] David Gibson, Jon Kleinberg, and Prabhakar Raghavan. 2000. Clustering categorical data: An approach based on dynamical systems. *The VLDB Journal—The International Journal on Very Large Data Bases* 8, 3-4 (2000), 222–236.

[31] David F. Gleich, Nate Veldt, and Anthony Wirth. 2018. Correlation Clustering Generalized. In *29th International Symposium on Algorithms and Computation (ISAAC 2018)*, Vol. 123. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 44:1–44:13. https://doi.org/10.4230/LIPIcs.ISAAC.2018.44

[32] Kaggle. 2015. Walmart Recruiting: Trip Type Classification. (2015). https://www.kaggle.com/c/walmart-recruiting-trip-type-classification.

[33] Kaggle. 2015. What's Cooking? (2015). https://www.kaggle.com/c/whats-cooking.

[34] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of the International Conference on Learning Representations*.

[35] Mikko Kivelä, Alex Arenas, Marc Barthelemy, James P. Gleeson, Yamir Moreno, and Mason A. Porter. 2014. Multilayer networks. *Journal of Complex Networks* 2, 3 (2014), 203–271.

[36] Pushmeet Kohli. 2007. *Minimizing Dynamic and Higher Order Energy Functions using Graph Cuts*. Ph.D. Dissertation. Oxford Brookes University.

[37] V. Kolmogorov and R. Zabin. 2004. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26, 2 (Feb 2004), 147–159. https://doi.org/10.1109/TPAMI.2004.1262177

[38] Andrea Lancichinetti and Santo Fortunato. 2012. Consensus clustering in complex networks. *Scientific Reports* 2, 1 (March 2012). https://doi.org/10.1038/srep00336

[39] Silvio Lattanzi and D Sivakumar. 2009. Affiliation networks. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*. Citeseer, 427–434.

[40] Yin Tat Lee and Aaron Sidford. 2015. Efficient inverse maintenance and faster algorithms for linear programming. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*. IEEE, 230–249.

[41] 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.

[42] Jure Leskovec, Kevin J. Lang, and Michael Mahoney. 2010. Empirical comparison of algorithms for network community detection. In *Proceedings of the 19th International Conference on World Wide Web - WWW '10*. ACM Press. https://doi.org/10.1145/1772690.1772755

[43] P. Li, H. Dau, G. Puleo, and O. Milenkovic. 2017. Motif clustering and overlapping clustering for social network analysis. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*. 1–9. https://doi.org/10.1109/INFOCOM.2017.8056956

[44] Pan Li and Olgica Milenkovic. 2017. Inhomogeneous Hypergraph Clustering with Applications. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 2308–2318. http://papers.nips.cc/paper/6825-inhomogeneous-hypergraph-clustering-with-applications.pdf

[45] Pan Li, Gregory J. Puleo, and Olgica Milenkovic. 2018. Motif and Hypergraph Correlation Clustering. *CoRR* abs/1811.02089 (2018). arXiv:1811.02089 http://arxiv.org/abs/1811.02089

[46] Cristopher Moore. 2017. The Computer Science and Physics of Community Detection: Landscapes, Phase Transitions, and Hardness. *Bulletin of the EATCS* 121 (2017).

[47] Peter J. Mucha, Thomas Richardson, Kevin Macon, Mason A. Porter, and Jukka-Pekka Onnela. 2010. Community structure in time-dependent, multiscale, and multiplex networks. *Science* 328, 5980 (2010), 876–878.

[48] Mark EJ Newman. 2003. The structure and function of complex networks. *SIAM Rev.* 45, 2 (2003), 167–256.

[49] James B Orlin. 2013. Max flows in $O(nm)$ time, or better. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. ACM, 765–774.

[50] Braxton Osting, Sourabh Palande, and Bei Wang. 2017. Spectral Sparsification of Simplicial Complexes for Clustering and Label Propagation. *arXiv:1708.08436* (2017).

[51] Pietro Panzarasa, Tore Opsahl, and Kathleen M Carley. 2009. Patterns and dynamics of users' behavior and interaction: Network analysis of an online community. *Journal of the American Society for Information Science and Technology* 60, 5 (2009), 911–932.

[52] Evangelos Papalexakis, Konstantinos Pelechrinis, and Christos Faloutsos. 2014. Spotting misbehaviors in location-based social networks using tensors. In *Proceedings of the 23rd International Conference on World Wide Web*. ACM, 551–552.

[53] Mason A Porter. 2019. Nonlinearity+ Networks: A 2020 Vision. *arXiv:1911.03805* (2019).
[54] Ryan A Rossi, Nesreen K Ahmed, and Eunyee Koh. 2018. Higher-order network representation learning. In *Companion Proceedings of the The Web Conference 2018*. 3–4.
[55] Vsevolod Salnikov, Daniele Cassese, and Renaud Lambiotte. 2018. Simplicial complexes and complex systems. *European Journal of Physics* 40, 1 (2018), 014001.
[56] Satu Elisa Schaeffer. 2007. Graph clustering. *Computer Science Review* 1, 1 (2007), 27–64.
[57] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June (Paul) Hsu, and Kuansan Wang. 2015. An Overview of Microsoft Academic Service (MAS) and Applications. In *WWW '15 Companion*. ACM.
[58] Substance Abuse and Mental Health Services Administration. Drug Abuse Warning Network (DAWN). 2011. (2011). https://www.samhsa.gov/data/data-we-collect/dawn-drug-abuse-warning-network.
[59] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. 2011. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proceedings of the VLDB Endowment* 4, 11 (2011), 992–1003.
[60] Chun-Yuen Teng, Yu-Ru Lin, and Lada A. Adamic. 2012. Recipe recommendation using ingredient networks. In *Proceedings of the 3rd Annual ACM Web Science Conference on - WebSci '12*. ACM Press. https://doi.org/10.1145/2380718.2380757
[61] Dorothea Wagner and Frank Wagner. 1993. Between min cut and graph bisection. In *International Symposium on Mathematical Foundations of Computer Science*. Springer, 744–750.
[62] Zhiqiang Xu, Yiping Ke, Yi Wang, Hong Cheng, and James Cheng. 2012. A model-based approach to attributed graph clustering. In *Proceedings of the 2012 International Conference on Management of Data*. ACM Press. https://doi.org/10.1145/2213836.2213894
[63] Hao Yin, Austin R. Benson, Jure Leskovec, and David F. Gleich. 2017. Local higher-order graph clustering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 555–564.
[64] Thomas Zaslavsky. 1982. Signed graphs. *Discrete Applied Mathematics* 4, 1 (1982), 47 – 74. https://doi.org/10.1016/0166-218X(82)90033-6
[65] Justine Zhang, Cristian Danescu-Niculescu-Mizil, Christina Sauper, and Sean J. Taylor. 2018. Characterizing Online Public Discussions through Patterns of Participant Interactions. *Proceedings of the ACM on Human-Computer Interaction* 2, CSCW (2018), 1–27. https://doi.org/10.1145/3274467
[66] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. 2007. Learning with hypergraphs: Clustering, classification, and embedding. In *Advances in Neural Information Processing Systems*. 1601–1608.
[67] Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. 2009. Graph clustering based on structural/attribute similarities. *Proceedings of the VLDB Endowment* 2, 1 (2009), 718–729. https://doi.org/10.14778/1687627.1687709