Journal of Combinatorial Optimization (2019) 37:1312–1326
https://doi.org/10.1007/s10878-018-0354-y

LP-based pivoting algorithm for higher-order correlation clustering

Takuro Fukunaga[1]

Published online: 22 October 2018

Abstract
Correlation clustering is an approach for clustering a set of objects from given pairwise information. In this approach, the given pairwise information is usually represented by an undirected graph with nodes corresponding to the objects, where each edge in the graph is assigned a nonnegative weight, and either the positive or negative label. Then, a clustering is obtained by solving an optimization problem of finding a partition of the node set that minimizes the disagreement or maximizes the agreement with the pairwise information. In this paper, we extend correlation clustering with disagreement minimization to deal with higher-order relationships represented by hypergraphs. We give two pivoting algorithms based on a linear programming relaxation of the problem. One achieves an $O(k \log n)$-approximation, where n is the number of nodes and k is the maximum size of hyperedges with the negative labels. This algorithm can be applied to any hyperedges with arbitrary weights. The other is an $O(r)$-approximation for complete r-partite hypergraphs with uniform weights. This type of hypergraphs arise from the coclustering setting of correlation clustering.

Keywords Correlation clustering · Coclustering · LP-rounding algorithm

1 Introduction

1.1 Problem formulation

In this paper, we consider approximation algorithms for the hypergraph correlation clustering. In the hypergraph correlation clustering, a problem instance consists of an

A preliminary version of this paper was published at the 24th International Computing and Combinatorics Conference (COCOON 2018).

✉ Takuro Fukunaga
takuro.fukunaga@riken.jp

[1] RIKEN Center for Advanced Intelligence Project, Tokyo, Japan

undirected hypergraph $G = (V, E)$ with the node set V and the hyperedge set E, and the label and the weight of each hyperedge in E. The label on a hyperedge is either positive or negative. We call a hyperedge positive if it is assigned the positive label, and negative otherwise. The sets of positive and negative hyperedges are denoted by E_+ and E_-, respectively (i.e., E is the disjoint union of E_+ and E_-). The weight of each hyperedge e is a nonnegative real number, denoted by $w(e)$.

The hypergraph correlation clustering is an optimization problem of finding a clustering of the given hypergraph $G = (V, E)$. A *clustering* $\mathcal{C}$ of G is defined as a partition of V into nonempty subsets. Each node set in $\mathcal{C}$ is called a *cluster*. A hyperedge e in G is defined to *disagree* with a clustering $\mathcal{C}$ if either of the following statements is true:

- e is a positive hyperedge, and some two end nodes of it belong to different clusters of $\mathcal{C}$;
- e is a negative hyperedge, and all of its end nodes belong to the same cluster of $\mathcal{C}$.

Then, the objective of the problem is to find a clustering minimizing the total weight of hyperedges that disagree with the clustering.

In a part of this paper, we focus on a special type of hypergraphs called *complete r-partite* hypergraphs. A hypergraph is called r-partite if its node set can be divided into disjoint r subsets $V_1, \ldots, V_r$ so that each hyperedge includes exactly one node from V_j for each $j = 1, \ldots, r$. An r-partite hypergraph is complete if each tuple $\{v_1, \ldots, v_r\} \in V_1 \times \cdots \times V_r$ is included as a hyperedge. We refer to the set of instances with complete r-partite hypergraphs and uniform hyperedge weights as *coclustering setting*; the reason for this name will be explained below.

1.2 Motivation

Correlation clustering is originally an approach for computing a clustering from given pairwise information. It was introduced by Bansal et al. (2004). They proposed representing the pairwise information as a graph with nodes corresponding to the objects to be clustered. Informations represented by the graph are possibly inconsistent due to existence of noise or observation errors. The purpose of correlation clustering is to find a clustering matching the pairwise information to the greatest degree possible. This purpose presents two optimization problems defined on the graph naturally; one seeks a clustering that minimizes the disagreement, and the other seeks a clustering that maximizes the agreement. Since these problems are NP-hard, several approximation algorithms have been proposed for them, and have been successfully applied to numerous applications in machine learning and computational biology (Ben-Dor et al. 1999; Cohen and Richman 2002; Filkov and Skiena 2003; McCallum and Wellner 2003). We will review these previous studies briefly in Sect. 2.

In several applications, pairwise information does not give enough information for the extraction of precise clusterings, and hence it is motivated to study clustering from higher-order information, which is modeled as the hypergraph correlation clustering. Even in the hypergraph correlation clustering, we can consider both the disagreement minimization and the agreement maximization. However, since this paper discusses only the disagreement minimization, we gave the disagreement minimization formulation above. A straightforward idea for the hypergraph correlation clustering is to

reduce the problem to the graph correlation clustering by expanding each hyperedge to some graphs like cliques. However, this idea does not give efficient algorithms as we will see in Sect. 3.2.

Study on the hypergraph correlation clustering was initiated by Kim et al. (2011) for an application to the image segmentation. Subsequently Kappes et al. (2016) and Kim et al. (2014) also considered the same problem. All of these studies are similar in that:

- they proposed linear programming (LP) relaxations for the hypergraph correlation clustering (with the disagreement minimization objective), and presented algorithms using LP solvers or tailored cutting-plane algorithms;
- when the solution output by the algorithms is not integer, they round it by a simple procedure of rounding up an arbitrary non-zero variable into an integer;
- they empirically showed that introducing higher-order relationships into correlation clustering improves the quality of image segmentation.

These observations indicate that efficient algorithms for the hypergraph correlation are useful in practice. On the other hand, to the best of our knowledge, no approximation algorithm with a provable performance guarantee is known for the problem. Motivated by this fact, our aim is to present performance guarantee of approximation algorithms for the hypergraph correlation clustering.

In addition to the general case of the hypergraph correlation clustering, we will study the coclustering setting of the problem. Coclustering denotes the task of clustering the objects which are categorized into two or more classes, and relationships of objects from different classes are considered. For example, this setting arises when we find a clustering of documents and words from word occurrences in documents. It is also known to be useful for clustering of gene expression data. To distinguish clustering from pairwise relationships and higher-order relationships, in this paper, we call the former by *biclustering*, and the latter by coclustering.

In correlation clustering, the biclustering setting implies that the given information is represented by bipartite graphs. This setting has been studied extensively (Ailon et al. 2012; Amit 2004; Chawla et al. 2015). These previous studies show that the disagreement minimization problem with complete bipartite graphs and uniform edge weights admits constant-factor approximation algorithms. In contrast, the coclustering setting has not been studied in the context of correlation clustering although it seems useful; for example, consider clustering users, search key words, and goods from purchase records in an E-commerce website; if a user i purchased a good j after searching with a key word s, then the category of i, j, and s are likely same, and hence solving the hypergraph correlation clustering defined from these order-3 relationships gives a more precise clustering rather than computing from pairwise relationships. We note that the hypergraph defined in this situation is 3-partite.

1.3 Contributions

We present two approximation algorithms with approximation guarantees for the hypergraph correlation clustering. One of the algorithms is for general hypergraphs. It has an $O(k \log n)$-approximation guarantee (Theorem 1), where n is the number

of nodes and k is the maximum size of hyperedges assigned the negative label. In other words, for any instance of the hypergraph correlation clustering, our algorithm outputs a clustering the objective function value of which is within a factor of $O(k \log n)$ from the optimal. The other algorithm is for the coclustering setting (the given hypergraph is complete r-partite and hyperedge weights are uniform). It achieves an $O(r)$-approximation guarantee (Theorem 2). Note that this approximation factor is a constant when r is a constant, and hence it extends the constant-approximation guarantees of Ailon et al. (2012), Amit (2004) and Chawla et al. (2015) for the disagreement minimization problem with complete bipartite graphs and uniform edge weights.

Our algorithms are *pivoting algorithms*, that compute a clustering by deciding a pivoting node and a cluster including it repeatedly. Most of the known approximation algorithms for correlation clustering are this type of algorithms. In our algorithms, the choice of a pivoting node and its cluster is based on an LP relaxation of the problem. Our LP relaxation is a straightforward extension of the one considered in Charikar et al. (2005), Chawla et al. (2015) and Demaine et al. (2006) for the disagreement minimization problem with graphs. Moreover, it is almost same as or simpler than those used in the previous studies Kappes et al. (2016) and Kim et al. (2011, 2014) on the hypergraph correlation clustering. Indeed, our algorithm works even with the LP relaxations considered in Kappes et al. (2016) and Kim et al. (2011, 2014), and hence it can replace the rounding algorithms therein.

In our $O(k \log n)$-approximation algorithm, we use the *region-growing* idea to define the cluster including a chosen pivoting node. Indeed, our algorithm generalizes the $O(\log n)$-approximation algorithms in Charikar et al. (2005) and Demaine et al. (2006) for graphs. On the other hand, our $O(r)$-approximation algorithm for the coclustering setting is based on a new idea. When $r = 2$, the coclustering setting is equivalent to the disagreement minimization on complete bipartite graphs with uniform weights. Although several constant-factor approximation algorithms are known for this case (Ailon et al. 2012; Amit 2004; Chawla et al. 2015), it seems difficult to extend them to $r \geq 3$ because they crucially relies on a structure of graphs representing inconsistent informations. Hence we design a new algorithm from scratch. It achieved a slightly worse approximation factor for $r = 2$ compared with the previous studies on the complete bipartite graphs.

1.4 Organization

The rest of this paper is organized as follows. Section 2 surveys related previous studies. Section 3 introduces notations, the LP relaxation used in our algorithms, and an outline of our algorithms. Section 3 also explains that reducing the hypergraph correlation problem to the graph correlation clustering is not efficient. Section 4 presents our $O(k \log n)$-approximation algorithm, and Sect. 5 gives our $O(r)$-approximation algorithm for the correlation clustering setting. Section 6 concludes the paper.

2 Related work

2.1 Correlation clustering

Both of the agreement maximization and the disagreement minimization formulations of correlation clustering were introduced by Bansal et al. (2004). Charikar et al. (2005) gave a factor 0.7664 approximation algorithm for the agreement maximization. For the disagreement minimization, Charikar et al. (2005) and Demaine et al. (2006) gave factor $O(\log n)$ approximations. Demaine et al. also proved that the disagreement minimization is equivalent to the minimum multicut problem. This equivalence indicates that obtaining a constant-factor approximation for the disagreement minimization is unique-games hard because of the hardness result on the minimum multicut problem given in Chawla et al. (2006).

Several special cases of the disagreement minimization also have been studied well. For example, Amit (2004), Ailon et al. (2012) and Chawla et al. (2015) considered the case where the graph is complete bipartite and weights are uniform. They gave constant-factor approximation algorithms for this case, and the current best approximate factor among them is 3 due to Chawla et al. (2015). Chawla et al. also considered complete graphs, and presented a 2.06-approximation algorithm for uniform weights, and 1.5-approximation algorithm for weights satisfying the triangle inequality.

Note that the above studies on correlation clustering all consider graphs. To the best of our knowledge, the correlation clustering over hypergraphs have been studied only in Kappes et al. (2016) and Kim et al. (2011, 2014), and no algorithm with a performance guarantee is known.

2.2 Coclustering

Biclustering of data represented by a matrix has been studied since the 1970s (Hartigan 1972). There has been a huge number of algorithms proposed so far, and we name a few of them Dhillon et al. (2003), Shan and Banerjee (2008) and Zha et al. (2001). These algorithms have been successfully applied to numerous unsupervised learning tasks (Chen et al. 2015; Zhu et al. 2015). In particular, clustering on gene expression data (Cheng and Church 2000; Madeira et al. 2010) and document classification (Bisson and Hussain 2008; Dhillon 2001; Hussain et al. 2010) are studied actively. Compared with biclustering, coclustering of higher-order relational data has not been extensively studied so far. Zhao and Zaki (2005) proposed a graph-based algorithm for coclustering. Hatano et al. (2017) proposed a coclustering algorithm based on sampling hypergraph multicuts. Other previous studies Papalexakis et al. (2013) and Peng and Li (2011) depend on an algebraic approach known as tensor rank decomposition.

3 Preliminaries

3.1 Notations

Let $G = (V, E)$ be a hypergraph with the node set V and the hyperedge set E. Throughout this paper, we let n denote the cardinality of V (i.e., $n = |V|$). The cardinality of a hyperedge e is called the *rank* of e, and the rank of a hypergraph G is defined as the maximum rank of hyperedges in G. Note that a hyperedge of rank 2 and a hypergraph of rank 2 are an edge and a graph, respectively. For $U \subseteq V$ and $H \subseteq E$, let $\delta(U; H)$ denote the set of hyperedges in H that include nodes both in U and $V \backslash U$, and $H[U]$ denote the set of hyperedges in H that include no nodes from $V \backslash U$. $G[U]$ denotes the sub-hypergraph of G induced by U (i.e., hypergraph with the node set U and the hyperedge set $E[U]$).

3.2 Reduction to the graph correlation clustering

A natural approach for solving the hypergraph correlation clustering is to apply an existing algorithm for the graph correlation clustering to the graph obtained by transforming the given hypergraph. However, the transformation of a hypergraph into a graph may change the structure of the problem drastically. To see this, suppose that a positive hyperedge of rank k' is replaced by a clique that consists of $\binom{k'}{2}$ positive edges. In the original hypergraph, the weight of a positive hyperedge is counted in the disagreement only once if the nodes in the hyperedge belong to more than one cluster. In contrast, in the corresponding graph, the contribution of the edges in the clique to the measured disagreement depends on how the clique is divided. For example, if all but one of the nodes in the clique belong to the same cluster, the weights of $k' - 1$ edges are counted, whereas if the nodes are all divided into different clusters, the weights of $\binom{k'}{2}$ edges are counted. Thus, the contributions are very different when the clique is divided into two clusters and when it is divided into k' clusters. Because of this fact, applying the best-known graph correlation clustering algorithm to the obtained graph only gives an $O(k' \log n)$-approximation even if all negative hyperedges in the given hypergraph are order-2, while our algorithm given in Sect. 4 attains an $O(\log n)$-approximation in this case. It seems hard to avoid this phenomenon even if we consider other ways of transformation. When the given hypergraph includes a negative hyperedge of order larger than 3, it seems difficult to bound the approximation factor given by the above approach; even if a clustering partitions a negative hyperedge into at least two clusters (and hence it incurs no cost from the hyperedge), an edge generated by transforming the negative hyperedge belongs to the same cluster in the clustering (and it incurs a positive cost).

3.3 Overview of our algorithms

In this subsection, we introduce our algorithms for the hypergraph correlation clustering. Our algorithms are based on an LP obtained by relaxing an integer programming (IP) formulation of the problem. We first introduce this IP formulation.

This formulation optimizes the following variables, which take numbers in $\{0, 1\}$:

- A variable x_{uv} for each pair of nodes $u, v \in V$; it is 0 if u and v belong to the same cluster, and it is 1 otherwise;
- A variable x_e for each hyperedge e; it is 0 if all nodes in e are included in a cluster, and it is 1 otherwise.

If a positive hyperedge $e \in E_+$ is included in a cluster, this implies that any two nodes u and v included in e belong to the same cluster. The following constraint formulates this condition:

$$x_{uv} \leq x_e, \ \forall e \in E_+, \forall \{u, v\} \subseteq e. \tag{1}$$

If a negative hyperedge $e = \{v_1, \ldots, v_r\} \in E_-$ intersects more than one cluster, then a node v_1 and some of the other nodes $v_2, \ldots, v_r$ belong to different clusters. This is represented by

$$x_e \leq \sum_{i=1}^{r-1} x_{v_i v_{i+1}}, \ \forall e = \{v_1, \ldots, v_r\} \in E_-. \tag{2}$$

Here, the ordering of nodes included in e is fixed arbitrarily.

If two nodes u and v belong to the same cluster, and if v and z also do, then all of these three nodes belong to the same cluster. This means that if $x_{uv} = x_{vz} = 0$, then $x_{uz} = 0$ must hold. Thus, the variables satisfy the following triangle inequalities:

$$x_{uz} \leq x_{uv} + x_{vz}, \ \forall u, v, z \in V. \tag{3}$$

Our IP formulation optimizes over these constraints. The disagreement objective function is $\sum_{e \in E_+} w(e) x_e + \sum_{e \in E_-} w(e)(1 - x_e)$. The LP relaxation is obtained from the IP formulation by relaxing the range of each variable to $[0, 1]$. Specifically, it is described as follows:

$$\begin{aligned} \text{minimize} \quad & \sum_{e \in E_+} w(e) x_e + \sum_{e \in E_-} w(e)(1 - x_e) \\ \text{subject to} \quad & (1), (2), (3), \\ & x_{uv} \in [0, 1], \quad \forall u, v \in V, \\ & x_e \in [0, 1], \quad \forall e \in E. \end{aligned} \tag{4}$$

For convenience, we let $x_{vv} = 0$ for all $v \in V$ in the rest of this paper although these variables do not appear in LP (4).

Our algorithms first compute an optimal solution x for the LP relaxation (4).

Then they construct a clustering from x in iterations. We let U refer to the set of nodes that belong to no cluster yet during the algorithm, where it is initialized to the V at the beginning of the algorithm. In each iteration, the algorithm computes a cluster by the following three steps:

(i) choose a node v from U (we call it *pivoting node*);

(ii) define the cluster containing v as $B_{x,v,U}(\xi) := \{u \in U : x_{uv} < \xi\}$ from some radius $\xi \in [0, 1]$;
(iii) remove the nodes in $B_{x,v,U}(\xi)$ from U and the hypergraph.

Selection of v in Step (i) and the definition of ξ used in Step (ii) are customized in two variations of our algorithms. Roughly speaking, we optimize them so that the ratio of weights of disagreed hyperedges incident to $B_{x,v,U}(\xi)$ to the fractional weights of hyperedges incident to $B_{x,v,U}(\xi)$ defined from x is minimized. Refer to Sects. 4 and 5 for the details. The algorithms are described in Algorithm 1.

Algorithm 1 LP-based pivoting algorithm for hypergraph correlation clustering

Input: a hypergraph $G = (V, E)$ with $E = E_+ \cup E_-$ and a nonnegative weight $w(e)$ for each $e \in E$
Output: a clustering $\mathcal{C}$ of V,
1: $\mathcal{C} \longleftarrow \emptyset$, $U \longleftarrow V$
2: compute an optimal solution x of (4)
3: **while** $U \neq \emptyset$ **do**
4: compute $v \in U$ and $\xi \in [0, 1]$ (details are described in Sections 4 and 5)
5: $\mathcal{C} \longleftarrow \mathcal{C} \cup \{B_{x,v,U}(\xi)\}$, $U \longleftarrow U \setminus B_{x,v,U}(\xi)$
6: remove all hyperedges intersecting $B_{x,v,U}(\xi)$ from E
7: **end while**
8: output $\mathcal{C}$

4 $O(k \log n)$-approximation for general hypergraphs

In this section, we discuss general hypergraphs with arbitrary hyperedge weights. First, let us introduce several notations. We let x and L refer to an optimal solution for (4) and its objective value. For a hyperedge e and a node v, let $d(e, v)$ and $d'(e, v)$ denote $\min_{u \in e} x_{vu}$ and $\max_{u \in e} x_{vu}$, respectively. For $\xi \in [0, 1]$, we define $F_{v,x,E}(\xi)$ and $C_{v,x,E}(\xi)$ by

$$F_{v,x,E}(\xi) := \frac{L}{n} + \sum_{e \in E_+[B_{x,v,U}(\xi)]} w(e) x_e + \sum_{e' \in \delta(B_{x,v,U}(\xi); E_+)} w(e') x_{e'} \frac{\xi - d(e', v)}{d'(e', v) - d(e', v)}$$

and

$$C_{v,x,E}(\xi) := \sum_{e \in \delta(B_{x,v,U}(\xi); E_+)} w(e),$$

where $C_{v,x,E}(\xi)$ is defined to be $+\infty$ if $\delta(B_{x,v,U}(\xi); E_+) = \emptyset$. We note that if $e' \in \delta(B_{x,v,U}(\xi); E_+)$, then $d(e', v) \leq \xi \leq d'(e', v)$ holds, and hence the third term in $F_{v,x,E}(\xi)$ is at most $\sum_{e' \in \delta(B_{v,x,U}(\xi); E)} w(e') x_{e'}$. Below, we omit the subscripts of $B_{x,v,U}(\xi)$, $F_{v,x,E}(\xi)$, and $C_{v,x,E}(\xi)$ when they are clear from the context. Roughly speaking, the second and the third terms of $F(\xi)$ represent how much the objective value of x in (4) is reduced when the positive hyperedges incident to $B(\xi)$ are removed

from the hypergraph, and $C(\xi)$ represents how much the disagreement of the positive hyperedges is increased when $B(\xi)$ is added as a cluster to a clustering.

Now, we are ready to describe details of our algorithm for general hypergraphs. In this variation, the pivoting node v is chosen arbitrarily from U. The radius ξ defining the cluster $B(\xi)$ including v is chosen from $[0, 1/(2k)]$ so that $C(\xi)/F(\xi)$ is minimized. Although this is a continuous optimization problem, it can be done in $O(n)$ evaluations of the objective because of the following reason. Call the nodes in U by $u_1, \ldots, u_{|U|}$ so that $x_{vu_1} \leq x_{vu_2} \cdots \leq x_{vu_{|U|}}$ holds. Let $i \in \{1, \ldots, |U| - 1\}$. For any $\xi', \xi'' \in (x_{vu_i}, x_{vu_{i+1}}]$ with $\xi' \leq \xi''$, we have $B(\xi') = B(\xi'')$, from which $F(\xi') \leq F(\xi'')$ and $C(\xi') = C(\xi'')$ follow. These two relationships indicate $C(\xi')/F(\xi') \geq C(\xi'')/F(\xi'')$. Therefore, the radius ξ minimizing $C(\xi)/F(\xi)$ can be found from $\{0, 1/(2k)\} \cup \{x_{vu} : u \in U, x_{vu} \leq 1/(2k)\}$, the size of which is $O(|U|) = O(n)$.

The approximation performance of our algorithm depends on $C(\xi)/F(\xi)$; if $C(\xi)/F(\xi) \leq \alpha$ for any iterations, it achieves $2\max\{k, \alpha\}$-approximation. Lemma 1 guarantees that there always exists a radius $\xi \in [0, 1/(2k)]$ such that $C(\xi)/F(\xi) \leq 2k \log(n + 1)$.

Lemma 1 *For any $v \in U$, there exists $\xi \in [0, 1/(2k)]$ such that $C_{v,x,E}(\xi) \leq 2k \log(n + 1) F_{v,x,E}(\xi)$.*

Proof Let $\xi \in (0, 1/(2k))$. $F(\xi)$ is differentiable unless $\xi = x_{vu}$ for some vertex $u \in V$. We let $a_1, \ldots, a_k \in (0, 1/(2k))$ be the numbers such that $0 < a_1 < a_2 < \cdots < a_k < 1/(2k)$ and $F(\xi)$ is not differentiable for any $\xi \in \{a_1, \ldots, a_k\}$. For notational convenience, we let a_0 and a_{k+1} denote 0 and $1/(2k)$.

Let $i \in \{0, \ldots, k\}$. Note that $x_e \geq d'(e, v) - d(e, v)$ holds because, if $u \in \arg\min_{u \in e} x_{vu}$ and $u' \in \arg\max_{u' \in e} x_{vu'}$ (i.e., $x_{vu} = d(e, v)$ and $x_{vu'} = d'(e, v)$), then $x_e \geq x_{uu'} \geq d'(e, v) - d(e, v)$, where the former inequality follows from (1) and the latter one follows from (3). Hence we have

$$\frac{d}{d\xi} F(\xi) = \sum_{e \in E_+ \cap \delta B(\xi)} w(e) \frac{x_e}{d'(e, v) - d(e, v)} \geq C(\xi)$$

for any $\xi \in (a_i, a_{i+1})$.

For notational convenience, we let $\epsilon = 2k \log(n + 1)$. To prove the lemma, we derive a contradiction under an assumption that $C(\xi) > \epsilon F(\xi)$ holds for all $\xi \in [0, 1/(2k)]$. From $\frac{d}{d\xi} F(\xi) > C(\xi)$ and $C(\xi) > \epsilon F(\xi)$, we have $\frac{d}{d\xi} F(\xi) > \epsilon F(\xi)$ for all $\xi \in (a_i, a_{i+1})$. This indicates that

$$\int_{F(a_i)}^{F(a_{i+1})} \frac{1}{F(\xi)} dF(\xi) > \int_{a_i}^{a_{i+1}} \epsilon = \epsilon(a_{i+1} - a_i).$$

Simultaneously, the left-hand side of this inequality is

$$\int_{F(a_i)}^{F(a_{i+1})} \frac{1}{F(\xi)} dF(\xi) = \log F(a_{i+1}) - \log F(a_i).$$

Therefore, we have

$$\log F\,(a_{i+1}) - \log F(a_i) > \epsilon(a_{i+1} - a_i).$$

Summing up this inequality over $i = 0, \ldots, k$ gives

$$\log F\left(\frac{1}{2k}\right) - \log F(0) > \frac{\epsilon}{2k}.$$

We observe that $F(0) = L/n$, and $F(1/(2k)) \leq (1+1/n)L$. Therefore, $\epsilon < 2k \log(n+1)$ holds, which is a contradiction. □

Theorem 1 *If the radius ξ is defined as in Lemma 1, the approximation factor of Algorithm 1 is $4k \log(n+1)$.*

Proof For each hyperedge $e \in E$, we define its LP value as $w(e)x_e$ if $e \in E_+$, and as $w(e)(1 - x_e)$ if $e \in E_-$. Notice that each hyperedge is removed from the hypergraph in some iteration of Algorithm 1. Let $E^{(i)}$ denote the set of hyperedges removed in an iteration i, and let $L^{(i)}$ denote the total LP value of hyperedges in $E^{(i)}$. We show that, for each iteration i, the total weight of disagreed hyperedges in $E^{(i)}$ is at most $2k \log(n+1)$ times $L^{(i)} + L/n$. This proves the theorem, because $\sum_i L^{(i)} \leq L$, the number of iterations is at most n, and L is at most the minimum disagreement of all clusterings.

Suppose that iteration i adds $B_{x,v,U}(\xi)$ to the output clustering $\mathcal{C}$. A hyperedge e is removed from the hypergraph in this iteration if and only if $e \cap B_{x,v,U}(\xi) \neq \emptyset$. A removed hyperedge e disagrees with the output clustering if one of the following holds: (a) e is negative and $e \subseteq B_{x,v,U}(\xi)$; (b) e is positive and $e \in \delta(B_{x,v,U}(\xi); E)$.

Let e be a negative hyperedge in $E^{(i)}$ that satisfies condition (a). Let r denote the order of e and, let $u_1, \ldots, u_r$ denote the nodes included in e. Since e is included in $B_{x,v,U}(\xi)$ and $\xi \leq 1/(2k)$, for any $i \in \{1, \ldots, r-1\}$, $x_{u_i u_{i+1}} \leq x_{vu_i} + x_{vu_{i+1}} \leq 1/k$ holds, where the former inequality follows from (3). By the constraint (2), $x_e \leq \sum_{i=1}^{r-1} x_{v_i v_{i+1}} \leq (r-1)/k \leq (k-1)/k$ holds, and hence $w(e)(1 - x_e) \geq w(e)/k$. Therefore, the weight of e is at most k times its LP value.

Notice that $C_{v,x,E}(\xi)$ is the total weight of positive hyperedges that satisfy condition (b). By Lemma 1, $C_{v,x,E}(\xi) \leq 2k \log(n+1) F_{v,x,E}(\xi)$ holds. Note that $F_{v,x,E}(\xi) - L/n$ is at most the total LP value of the removed positive hyperedge. Therefore, the required claim is proven. □

As mentioned in Sect. 2, for the disagreement minimization on graphs, the best known approximation factor is $O(\log n)$ (Charikar et al. 2005; Demaine et al. 2006), and the approximation factor of our algorithm matches it up to a constant when the problem is restricted to graphs. It is an obvious open problem to improve this factor, even for graphs. It is unique-games hard to obtain a constant-factor approximation algorithm for graphs (Demaine et al. 2006). Since hypergraphs include graphs, the same hardness result applied to the hypergraph correlation clustering.

5 $O(r)$-approximation for the coclustering setting

In this section, we consider the coclustering setting. In other words, the node set of the input hypergraph is the disjoint union of $V_1, \dots, V_r$, the set of hyperedges coincides with $V_1 \times V_2 \times \cdots \times V_r$, and each hyperedge is associated with a unit weight. The task is to find a partition $\mathcal{C}$ of $\bigcup_{i=1}^{k} V_i$ that minimizes $|\{e \in E_+ : e \in \delta(\mathcal{C})\}| + |\{e \in E_- : e \in E(\mathcal{C})\}|$.

In our pivoting algorithm for this case, the ordering of nodes used for defining the constraint (2) is decided by an ordering of $V_1, \dots, V_r$. For each $j = 1, \dots, r$, let v_{i_j} be the node in $V_j \cap e$ for a negative hyperedge $e \in E_-$. Then, (2) demands that $x_e \leq \sum_{i=1}^{r-1} x_{v_i v_{i+1}}$ holds.

Our algorithm chooses the pivoting node v from $U \cap V_1$ in a certain way whenever $U \cap V_1 \neq \emptyset$, and the radius ξ is set to $1/\sqrt{2(r-1)(2r-1)}$. When $U \cap V_1 = \emptyset$, the clustering of the remaining nodes in U makes no effect on the objective value of the solution because no hyperedge remains in the hypergraph. Hence the algorithm stops the iterations and terminates after adding an arbitrary clustering of the remaining nodes to the solution.

To describe the choice of the pivoting node, let us introduce notations. Let $\mathbf{1}$ be the indicator function for events; if an even $\mathcal{E}$ happens, then $\mathbf{1}(\mathcal{E}) = 1$, and $\mathbf{1}(\mathcal{E}) = 0$ otherwise. In what follows, we rewrite the radius ξ as $1/\theta$ for notational convenience, and assume that θ is a fixed parameter; later, we show that $\theta = \sqrt{2(r-1)(2r-1)}$ minimizes the approximation factor. In each iteration, for a node $v \in U$ and a remaining hyperedge e, we define two costs $L_v(e)$ and $A_v(e)$ as follows:

$$L_v(e) = \begin{cases} x_e \cdot \mathbf{1}(B(v, 1/\theta) \cap e \neq \emptyset) & \text{if } e \in E_+, \\ (1 - x_e) \cdot \mathbf{1}(B(v, 1/\theta) \cap e \neq \emptyset) & \text{if } e \in E_-, \end{cases}$$

$$A_v(e) = \begin{cases} \mathbf{1}(B(v, 1/\theta) \cap e \neq \emptyset \neq e \setminus B(v, 1/\theta)) & \text{if } e \in E_+, \\ \mathbf{1}(e \subseteq B(v, 1/\theta)) & \text{if } e \in E_-. \end{cases}$$

If v is chosen as a pivoting node in this iteration, the cost of the LP solution is decreased by $\sum_{e \in E_+ \cup E_-} L_v(e)$, and the cost of the solution is increased by $\sum_{e \in E_+ \cup E_-} A_v(e)$. In our algorithm, in each iteration, we choose a node $v \in V_1 \cap U$ minimizing $\sum_{e \in E_+ \cup E_-} A_v(e) / \sum_{e \in E_+ \cup E_-} L_v(e)$ as the pivoting node.

If the pivoting node v satisfies $\sum_{e \in E_+ \cup E_-} A_v(e) / \sum_{e \in E_+ \cup E_-} L_v(e) \leq \alpha$ for some $\alpha \geq 1$ in each iteration, then it can be proven that the algorithm achieves α-approximation. Below, we prove that the condition is satisfied with

$$\begin{aligned} \alpha &= \frac{2r-2}{\sqrt{2(r-1)(2r-1)} - 2r + 2} - \frac{2r-1}{\sqrt{2(r-1)(2r-1)} - 2r + 1} \\ &= 2\sqrt{2(r-1)(2r-1)} + 4r - 3 \end{aligned} \tag{5}$$

when $\theta = \sqrt{2(r-1)(2r-1)}$. Indeed, we prove that

$$\sum_{v \in V_1 \cap U} \sum_{e \in E_+ \cup E_-} A_v(e) \leq \alpha \sum_{v \in V_1 \cap U} \sum_{e \in E_+ \cup E_-} L_v(e) \tag{6}$$

holds with α satisfying (5). Notice that this implies that the node chosen as the pivoting node satisfies the required condition.

In the rest of this section, we prove (6) under an assumption that $U = V$; if $U \subset V$, (6) is proven by applying the following discussion to the sub-hypergraph induced by U. First, we bound $\sum_{v\in V_1}\sum_{e\in E_-} A_v(e)$ in the following lemma.

Lemma 2 $\sum_{v\in V_1}\sum_{e\in E_-} A_v(e) \le \frac{\theta}{\theta-2r+2}\sum_{v\in V_1}\sum_{e\in E_-} L_v(e)$.

Proof Let $e = i_1 i_2 \cdots i_r \in E_-$. $A_v(e) = 1$ whenever $e \subseteq B(v, 1/\theta)$. In this case, $x_{i_j i_{j+1}} \le 1/\theta$ holds for all $j = 1, \ldots, r-1$, which indicates $x(e) \le \sum_{j=1}^{r-1} x_{i_j i_{j+1}} \le x_{vi_1} + 2\sum_{j=2}^{r-1} x_{vi_j} + x_{vi_r} \le 2(r-1)/\theta$. This means that $L_v(e) = 1 - x_e \ge (1 - 2(r-1)/\theta) \cdot A_v(e)$ in this case. The same inequality trivially holds when $A_v(e) = 0$. Therefore, $\sum_{v\in V_1}\sum_{e\in E_-} A_v(e) \le \frac{\theta}{\theta-2r+2}\sum_{v\in V_1}\sum_{e\in E_-} L_v(e)$. □

Next, we bound $\sum_{v\in V_1}\sum_{e\in E_+} A_v(e)$. We introduce a parameter β that satisfies $0 \le \beta \le 1/\theta$. Let us remark that $1 - 1/\theta - (r-1)\beta \ge 0$ holds because $1/\theta \le (1 - 1/\theta)/(r-1)$ follows from $\theta \ge 2r-2 \ge r$. We first bound $\sum_{v\in V_1}\sum_{e\in E_+:x(e)\ge\beta} A_v(e)$.

Lemma 3 $\sum_{v\in V_1}\sum_{e\in E_+:x(e)\ge\beta} A_v(e) \le \frac{1}{\beta}\sum_{v\in V_1}\sum_{e\in E_+} L_v(e)$.

Proof Let $e \in E_+$ such that $A_v(e) = 1$. Then $L_v(e) = x_e$. If $x(e) \ge \beta$, this means that $\beta A_v(e) \le L_v(e)$ holds. □

Lemma 4 $\sum_{v\in V_1}\sum_{e\in E_+:x(e)<\beta} A_v(e) \le \frac{\theta}{1-\theta\beta}\sum_{v\in V_1}\sum_{e\in E_+\cup E_-} L_v(e)$.

Proof Let $e = \{i_1 \cdots i_r\} \in E_+$ such that $A_v(e) = 1$ and $x(e) < \beta$. $A_v(e) = 1$ indicates that $e \cap B(v, 1/\theta) \ne \emptyset \ne e \backslash B(v, 1/\theta)$. In other words, there exist $s, t \in \{i_1, \ldots, i_r\}$ such that $x_{vs} \le 1/\theta < x_{vt}$. Without loss of generality, let $i_1 \in V_1$. Let $e' = vi_2 \cdots i_r$ ($e = e'$ when $v = i_1$, and the following discussion holds even in this case). We bound $A_v(e)$ by $L_{i_1}(e')$. Since $x_{i_1 i_2} \le x(e) < \beta \le 1/\theta$, we have $e' \cap B(i_1, 1/\theta) \ne \emptyset$, indicating that $L_{i_1}(e') = x_{e'}$ if $e' \in E_+$ and $L_{i_1}(e') = 1 - x_{e'}$ if $e' \in E_-$.

We first consider the case of $e' \in E_+$. Since $r \ge 3$, there exists at least one node in $\{i_2, \ldots, i_r\}$ distinct from t. Without loss of generality, let i_2 be such a node (if $i_2 = t$, exchange the subscripts of i_2 and i_3). Then, $L_{i_1}(e') = x_{e'} \ge x_{vi_2} \ge x_{vt} - x_{ti_2} > 1/\theta - \beta$, where the last inequality follows from $x_{vt} > 1/\theta$ and $x_{ti_2} \le x_e \le \beta$.

Next, we consider the case of $e' \in E_-$. In this case, we assume that $\{i_j\} = e \cap V_j$ for each $j = 2, \ldots, r$, and hence the constraint (2) corresponding to e' demands that $x_{e'} \le x_{vi_2} + \sum_{j=2}^{r-1} x_{i_j i_{j+1}}$ holds. The second term x_{vi_2} in the right-hand side of this inequality is bounded as $x_{vi_2} \le x_{vs} + x_{si_2}$ by (3). Recall that $x_{vs} \le 1/\theta$ holds by the definition of s. Moreover, $x_{si_2} \le x(e) \le \beta$ holds because both s and i_2 are included in e and the inequality follows from (1) if $s \ne i_2$ (the inequality is immediate if $s = i_2$). Thus, we have $x_{vi_2} \le 1/\theta + \beta$. Moreover, for any $j \in \{2, \ldots, r-1\}$, we have $x_{i_j i_{j+1}} \le \beta$. Therefore, we have $x_{e'} \le 1/\theta + (r-1)\beta$. Then, $L_{i_1}(e') = 1 - x_{e'} \ge 1 - 1/\theta - (r-1)\beta$ holds. We note that $1 - 1/\theta - (r-1)\beta \ge 1/\theta - \beta$ follows from $\beta \le 1/\theta$ and $\theta \ge 2r - 2 \ge r$.

Summing up, in either case, $L_{i_1}(e') \geq 1/\theta - \beta$ holds. Hence, $L_{i_1}(e')$ can be used to bound $A_v(e)$. Let us notice that $L_{i_1}(e')$ is not used for bounding more than one pair of v and e because (i_1, e') is uniquely determined from (v, e). Therefore, we have

$$\begin{aligned}\sum_{v\in V_1}\sum_{e\in E_+ : x(e)<\beta} A_v(e) &= \sum_{v\in V_1}\sum_{e\in E_+ : x(e)<\beta} \mathbf{1}(A_v(e)=1)\\ &\leq \frac{\theta}{1-\theta\beta}\sum_{v\in V_1}\sum_{e\in E_+ : x(e)<\beta} L_{i_1}(e')\\ &\leq \frac{\theta}{1-\theta\beta}\sum_{i_1\in V_1}\sum_{e'\in E_+\cup E_-} L_{i_1}(e').\end{aligned}$$

□

From Lemmas 2, 3, and 4, we obtain the following inequality:

$$\sum_{v\in V_1}\sum_{e\in E_+\cup E_-} A_v(e) \leq \left(\max\left\{\frac{\theta}{\theta-2r+2}, \frac{1}{\beta}\right\} + \frac{\theta}{1-\theta\beta}\right)\sum_{v\in V_1}\sum_{e\in E_+\cup E_-} L_v(e).$$

Hence, (6) is satisfied when α is the minimum value of $\max\{\theta/(\theta-2r+2), 1/\beta\} + \theta/(1-\theta\beta)$ subject to $\theta \geq 2r-2$ and $0 \leq \beta \leq 1/\theta$. This minimum value is equal to the right-hand side of (5), which is attained by $\theta = \sqrt{2(r-1)(2r-1)}$ and $\beta = 1 - \sqrt{2(r-1)/(2r-1)}$.

Theorem 2 *If $\xi = 1/\sqrt{2(r-1)(2r-1)}$ and the pivoting node v is the one in $U \cap V_1$ minimizing $\sum_{e\in E_+\cup E_-} A_v(e) / \sum_{e\in E_+\cup E_-} L_v(e)$, then the approximation factor of Algorithm 1 is $2\sqrt{2(r-1)(2r-1)} + 4r - 3$ for the coclustering setting.*

6 Conclusion

We considered the hypergraph correlation clustering, and gave two approximation guarantees for the LP-based pivoting algorithms. One is an $O(k \log n)$-approximation guarantee, and the other is an $O(r)$-approximation guarantee. In practice, the former guarantee is more useful because it deals with arbitrary weights while the latter is restricted to coclustering setting. Nevertheless, the latter guarantee is interesting in relationship with previous studies on the disagreement minimization with bipartite graphs (Ailon et al. 2012; Amit 2004; Chawla et al. 2015).

Funding Funding was provided by Japan Society for the Promotion of Science (Grant No. JP17K00040).

References

Ailon N, Avigdor-Elgrabli N, Liberty E, van Zuylen A (2012) Improved approximation algorithms for bipartite correlation clustering. SIAM J Comput 41(5):1110–1121

Amit N (2004) The bicluster graph editing problem. Master's thesis, Tel Aviv University

Bansal N, Blum A, Chawla S (2004) Correlation clustering. Mach Learn 56(1–3):89–113

Ben-Dor A, Shamir R, Yakhini Z (1999) Clustering gene expression patterns. J Comput Biol 6(3/4):281–297

Bisson G, Hussain SF (2008) Chi-sim: a new similarity measure for the co-clustering task. In: Proceedings of the seventh international conference on machine learning and applications, ICMLA 2008, pp 211–217

Charikar M, Guruswami V, Wirth A (2005) Clustering with qualitative information. J Comput Syst Sci 71(3):360–383

Chawla S, Krauthgamer R, Kumar R, Rabani Y, Sivakumar D (2006) On the hardness of approximating multicut and sparsest-cut. Comput Complex 15(2):94–114

Chawla S, Makarychev K, Schramm T, Yaroslavtsev G (2015) Near optimal LP rounding algorithm for correlation clustering on complete and complete k-partite graphs. In: Proceedings of the forty-seventh annual ACM on symposium on theory of computing, STOC 2015, pp 219–228

Chen X, Ritter A, Gupta A, Mitchell TM (2015) Sense discovery via co-clustering on images and text. In: Proceedings of IEEE conference on computer vision and pattern recognition, CVPR 2015, pp 5298–5306

Cheng Y, Church GM (2000) Biclustering of expression data. In: Proceedings of the eighth international conference on intelligent systems for molecular biology, pp 93–103

Cohen WW, Richman J (2002) Learning to match and cluster large high-dimensional data sets for data integration. In: Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining, pp 475–480

Demaine ED, Emanuel D, Fiat A, Immorlica N (2006) Correlation clustering in general weighted graphs. Theor Comput Sci 361(2–3):172–187

Dhillon IS (2001) Co-clustering documents and words using bipartite spectral graph partitioning. In: Proceedings of the seventh ACM SIGKDD international conference on knowledge discovery and data mining, pp 269–274

Dhillon IS, Mallela S, Modha DS (2003) Information-theoretic co-clustering. In: Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining, pp 89–98

Filkov V, Skiena S (2003) Integrating microarray data by consensus clustering. In: Proceedings of the 15th IEEE international conference on tools with artificial intelligence, pp 418–425

Hartigan JA (1972) Direct clustering of a data matrix. J Am Stat Assoc 67(337):123–129

Hatano D, Fukunaga T, Kawarabayashi K (2017) Scalable algorithm for higher-order co-clustering via random sampling. In: Proceedings of the thirty-first AAAI conference on artificial intelligence, 4–9 Feb 2017, San Francisco, California, USA, pp 1992–1999

Hussain SF, Bisson G, Grimal C (2010) An improved co-similarity measure for document clustering. In: Proceedings of the ninth international conference on machine learning and applications, ICMLA 2010, pp 190–197

Kappes JH, Speth M, Reinelt G, Schnörr C (2016) Higher-order segmentation via multicuts. Comput Vis Image Underst 143:104–119

Kim S, Nowozin S, Kohli P, Yoo CD (2011) Higher-order correlation clustering for image segmentation. In: Advances in neural information processing systems 24: proceedings of the 25th annual conference on neural information processing systems 2011, pp 1530–1538

Kim S, Yoo CD, Nowozin S, Kohli P (2014) Image segmentation usinghigher-order correlation clustering. IEEE Trans Pattern Anal Mach Intell 36(9):1761–1774

Madeira SC, Teixeira MC, Sá-Correia I, Oliveira AL (2010) Identification of regulatory modules in time series gene expression data using a linear time biclustering algorithm. IEEE ACM Trans Comput Biol Bioinform 7(1):153–165

McCallum A, Wellner B (2003) Toward conditional models of identity uncertainty with application to proper noun coreference. In: Proceedings of IJCAI-03 workshop on information integration on the web, IIWeb-03, pp 79–84

Papalexakis EE, Sidiropoulos ND, Bro R (2013) From K-means to higher-way co-clustering: multilinear decomposition with sparse latent factors. IEEE Trans Signal Process 61(2):493–506

Peng W, Li T (2011) Temporal relation co-clustering on directional social network and author-topic evolution. Knowl Inf Syst 26(3):467–486

Shan H, Banerjee A (2008) Bayesian co-clustering. In: Proceedings of the 8th IEEE international conference on data mining, ICDM 2008, pp 530–539

Zha H, He X, Ding CHQ, Gu M, Simon HD (2001) Bipartite graph partitioning and data clustering. In: Proceedings of the 2001 ACM CIKM international conference on information and knowledge management, pp 25–32

Zhao L, Zaki MJ (2005) Tricluster: an effective algorithm for mining coherent clusters in 3D microarray data. In: Proceedings of the ACM SIGMOD international conference on management of data, pp 694–705

Zhu Y, Yang H, He J (2015) Co-clustering based dual prediction for cargo pricing optimization. In: Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, pp 1583–1592