Chromatic Correlation Clustering

FRANCESCO BONCHI, Yahoo Labs
ARISTIDES GIONIS, Aalto University
FRANCESCO GULLO, Yahoo Labs
CHARALAMPOS E. TSOURAKAKIS, Harvard School of Engineering and Applied Sciences
ANTTI UKKONEN, Aalto University

We study a novel clustering problem in which the pairwise relations between objects are *categorical*. This problem can be viewed as clustering the vertices of a graph whose edges are of different types (*colors*). We introduce an objective function that ensures the edges within each cluster have, as much as possible, the same color. We show that the problem is **NP**-hard and propose a randomized algorithm with approximation guarantee proportional to the maximum degree of the input graph. The algorithm iteratively picks a random edge as a pivot, builds a cluster around it, and removes the cluster from the graph. Although being fast, easy-to-implement, and parameter-free, this algorithm tends to produce a relatively large number of clusters. To overcome this issue we introduce a variant algorithm, which modifies how the pivot is chosen and how the cluster is built around the pivot. Finally, to address the case where a fixed number of output clusters is required, we devise a third algorithm that directly optimizes the objective function based on the *alternating-minimization* paradigm.

We also extend our objective function to handle cases where object's relations are described by multiple labels. We modify our randomized approximation algorithm to optimize such an extended objective function and show that its approximation guarantee remains proportional to the maximum degree of the graph

We test our algorithms on synthetic and real data from the domains of social media, protein-interaction networks, and bibliometrics. Results reveal that our algorithms outperform a baseline algorithm both in the task of reconstructing a ground-truth clustering and in terms of objective-function value.

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Applications - *Data Mining*

General Terms: Algorithms, Theory, Performance, Experimentation

Additional Key Words and Phrases: Clustering, Edge-labeled graphs, Correlation Clustering

ACM Reference Format:
Francesco Bonchi, Aristides Gionis, Francesco Gullo, Charalampos E. Tsourakakis, Antti Ukkonen, 2015. Chromatic Correlation Clustering. *ACM Trans. Knowl. Discov. Data.* V, N, Article A (January 2015), 20 pages.
DOI:http://dx.doi.org/10.1145/0000000.0000000

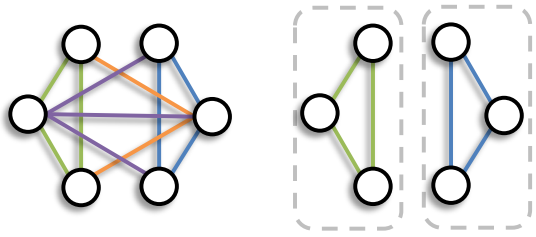
1. INTRODUCTION

Clustering is one of the most fundamental and well-studied problems in data mining. The goal of clustering is to partition a set of objects in different clusters, so that objects in the same cluster are more similar to each other than to objects in other clusters. A common trait underlying most clustering paradigms is the existence of a real-valued proximity function $f(\cdot,\cdot)$ representing the similarity/distance between pairs of objects. The proximity function is either provided explicitly as input, or it can be computed implicitly from the representation of the objects.

In this paper we consider a clustering setting where the relation among objects is represented by a relation type, that is a label from a finite set of possible labels L, while a special label $l_0 \notin L$ is used to denote that the objects have no relation. In other words,

Author's addresses: F. Bonchi and F. Gullo, Yahoo Labs, Spain, Email: {bonchi, gullo}@yahoo-inc.com; A. Gionis and A. Ukkonen, Helsinki Institute for Information Technology (HIIT), Aalto University, Finland, Email: {aristides.gionis, antti.ukkonen}@aalto.fi; C. E. Tsourakakis, Harvard School of Engineering and Applied Sciences, USA, Email: babis@seas.harvard.edu.
Most of the work was done while all the authors were at Yahoo Labs Barcelona. C. E. Tsourakakis was supported by the Yahoo! Internship program.

DOI:http://dx.doi.org/10.1145/0000000.0000000



(a) (b)

Fig. 1: An example of chromatic clustering: (a) input graph, (b) the optimal solution for CHROMATIC-CORRELATION-CLUSTERING (Problem 2).

we consider the case where the range of the proximity function $f(\cdot,\cdot)$ is *categorical*, instead of *numerical*. This setting has a natural graph interpretation: the input can be viewed as an *edge-labeled graph* $G = (V, E, L, l_0, \ell)$, where the set of vertices V corresponds to the objects to be clustered, the set of edges E comprises all unordered pairs within V having some relation (i.e., whose relation is represented by a label other than the l_0 label), and the function ℓ assigns to each edge in E a label from L. We find it intuitive to think of the edge label as the "color" of that edge, we thus use the terms *label* and *color* (as well as the terms *edge-labeled graph* and *chromatic graph*) interchangeably throughout the paper.

The key objective in our framework is to find a partition of the vertices of the graph such that the edges in each cluster have, as much as possible, the same color (label). An example is shown in Figure 1. Intuitively, a `red` edge (x, y) provides positive evidence that the vertices x and y should be clustered in such a way that the edges in the subgraph induced by that cluster are mostly `red`. Furthermore, in the case that most edges of a cluster are `red`, it is reasonable to label the whole cluster with the `red` color. Note that a clustering algorithm for this problem should also deal with inconsistent evidence, as a `red` edge (x, y) provides evidence for the vertex x to participate in a cluster with `red` edges, while a `green` edge (x, z) provides contradicting evidence for the vertex x to participate in a cluster with `red` edges. Aggregating such inconsistent information is resolved by optimizing an objective function that takes into account such designing principles.

Applications. The study of edge-labeled graphs is motivated by many real-world applications and is receiving increasing attention in the data-mining literature [Fan et al. 2011; Jin et al. 2010; Xu et al. 2011]. As an example, social networks are commonly represented as graphs, where the vertices represent individuals and the edges model relationships among these individuals. Again, these relationships can be of various types, e.g., colleagues, neighbors, schoolmates, football-mates.

As a further example, biologists study protein-protein interaction networks, where vertices represent proteins and edges represent interactions occurring when two or more proteins bind together to carry out their biological function. Those interactions can be of different types, e.g., physical association, direct interaction, co-localization, etc. In these networks, for instance, a cluster containing mainly edges labeled as co-localization, might represent a *protein complex*, i.e., a group of proteins that interact with each other at the same time and place, forming a single multi-molecular machine [Lin et al. 2007].

In bibliographic data, co-authorship networks represent collaborations among authors: in this case the topic of the collaboration can be viewed as an edge label, and a cluster represents a topic-coherent community of researchers.

In our experiments in Section 5 we show how our framework applies to all the above domains.

Contributions. In this paper we study the problem of clustering objects with categorical similarity. The problem was originally introduced in [Bonchi et al. 2012]. Here we present an extended version of that work. Our contributions can be summarized as follows.

- We define CHROMATIC-CORRELATION-CLUSTERING, a novel clustering problem for objects with categorical similarity, by revisiting the well-studied *correlation-clustering* framework [Bansal et al. 2004]. We show that our problem is a generalization of the traditional CORRELATION-CLUSTERING problem, implying that it is **NP**-hard.

- We extend our problem to address the case where relations between objects are described by a *set* of labels, rather than a single label only; this case naturally arises in variety of scenarios. In this respect, we define a generalized version of our problem, which we call MULTI-CHROMATIC-CORRELATION-CLUSTERING.
- We introduce a randomized algorithm, named Chromatic Balls, that provides an approximation guarantee proportional to the maximum degree of the graph.
- Though of theoretical interest, Chromatic Balls has some limitations when it comes to practice. Trying to overcome these limitations, we introduce two alternative algorithms: a more practical *lazy* version of Chromatic Balls, and an algorithm that directly optimizes the proposed objective function via an iterative process based on the *alternating-minimization* paradigm.
- We introduce Multi-Chromatic Balls, a modified version of our Chromatic Balls approximation algorithm to handle MULTI-CHROMATIC-CORRELATION-CLUSTERING problem. We show that Multi-Chromatic Balls still achieves an approximation guarantee: event though the approximation factor increases by a factor equal to size of the input label set, it still remains remains proportional to the maximum degree in the graph.
- We empirically evaluate our algorithms both on synthetic and real datasets. Experiments on synthetic data show that all our algorithms outperform a baseline algorithm in the task of reconstructing a ground-truth clustering. Experiments on real-world data confirm high accuracy in terms of objective-function values.

Roadmap. The rest of the paper is organized as follows. In the next section we recall the traditional correlation clustering problem and introduce our new formulations, for both the single-label case and multiple-label case. In Section 3 we describe our proposed algorithms for the single-label version of the problem: the Chromatic Balls algorithm and its approximation guarantee, along with two more practical algorithms, namely Lazy Chromatic Balls and Alternating Minimization. In Section 4 we introduce our Multi-Chromatic Balls for the multiple-label formulation. In Section 5 we report our experimental analysis. In Section 6 we discuss related work, while Section 7 concludes the paper.

2. PROBLEM DEFINITION

Background: correlation clustering. Generally speaking, a clustering problem asks us to partition a set of objects into groups (clusters) of similar objects. Let V be a set of objects and let V_2 denote the set of all *unordered pairs* of objects in V, i.e., $V_2 = \{S \subseteq V \ : \ |S| = 2\}$. Let also $sim : V_2 \to \mathbb{R}$ be a real-valued pairwise similarity function defined over the objects in V; typically the range of sim is $[0, 1]$. Assuming that cluster identifiers are represented by natural numbers, a clustering $\mathcal{C}$ can be viewed as a function $\mathcal{C} : V \to \mathbb{N}$. Typically, the goal is to find a clustering $\mathcal{C}$ that optimizes an objective function that measures the quality of the clustering based on the function $sim(\cdot, \cdot)$. Numerous formulations and objective functions have been defined in the literature. One of these, considered in both the area of theoretical computer science and data mining, is the foundation of the CORRELATION-CLUSTERING problem [Bansal et al. 2004].

PROBLEM 1 (CORRELATION-CLUSTERING). *Given a set of objects V and a pairwise similarity function $sim : V_2 \to [0, 1]$, find a clustering $\mathcal{C} : V \to \mathbb{N}$ that minimizes the cost*

$$\text{cost}(V, sim, \mathcal{C}) = \sum_{\substack{(x,y)\in V_2,\\ \mathcal{C}(x)=\mathcal{C}(y)}} (1 - sim(x, y)) \ + \sum_{\substack{(x,y)\in V_2,\\ \mathcal{C}(x)\neq\mathcal{C}(y)}} sim(x, y). \quad \square \tag{1}$$

The intuition underlying the above problem is that the cost of placing two objects x and y in the same cluster should be equal to the dissimilarity $1 - sim(x, y)$, while the cost of having the objects in different clusters should correspond to their similarity $sim(x, y)$. A common scenario is when the similarity is binary, that is $sim : V_2 \to \{0, 1\}$. In this case Equation (1) reduces to counting the (unordered) pairs of objects that have similarity 0 and are put in the same cluster plus the number of pairs of objects that have similarity 1 and belong to different clusters.

The CORRELATION-CLUSTERING problem, both the general and the binary version, can alternatively be formulated from a graph perspective. Considering for example binary CORRELATION-CLUSTERING, the input to the problem can be represented as a graph where the set of vertices corresponds to the set of objects to be clustered, while two objects are adjacent

if and only if they have similarity 1. Then, the objective function simply counts the number of edges that are cut plus the number of absent edges (i.e., unconnected unordered pairs of objects) that are not cut.

We show next how to extend the CORRELATION-CLUSTERING formulation to the case where the relations between objects are categorical. We first consider single-label relations (Section 2.1); then, we focus on the more general case where relations are described by multiple labels (Section 2.2).

2.1. Chromatic correlation clustering

The formulation of our clustering problem where objects have categorical relations resembles the CORRELATION-CLUSTERING formulation with binary similarities. Particularly, adopting a graph-based terminology, the input to our problem is an edge-labeled graph G, whose edges have a color (label) taken from a set L. We also assume that every absent edge in G is implicitly labeled with a special label $l_0 \notin L$. The objective is to derive a partition of the vertices V in G (i.e., a clustering $\mathcal{C} : V \to \mathbb{N}$) such that the relations among vertices in the same cluster are as much color-homogeneous as possible, while minimizing, at the same time, the number of edges across different clusters. We also want to characterize each output cluster with the label that best represents the edges therein; thus, our output also comprises a cluster labeling function $c\ell : \mathcal{C}[V] \to L$ (where $\mathcal{C}[V]$ denotes the codomain of $\mathcal{C}$) that assigns a label from L to each cluster. The formal definition of our problem, which we call CHROMATIC-CORRELATION-CLUSTERING, is reported next.

PROBLEM 2 (CHROMATIC-CORRELATION-CLUSTERING). *Let $G = (V, E, L, l_0, \ell)$ be an edge-labeled graph, where V is a set of vertices, $E \subseteq V_2$ is a set of edges, L is a set of labels, $l_0 \notin L$ is a* special *label, and $\ell : V_2 \to L \cup \{l_0\}$ is a labeling function that assigns a label to each unordered pair of vertices in V such that $\ell(x, y) = l_0$ if and only if $(x, y) \notin E$. We want to find a clustering $\mathcal{C} : V \to \mathbb{N}$ and a cluster labeling function $c\ell : \mathcal{C}[V] \to L$ so as to minimize the cost*

$$\text{cost}(G, \mathcal{C}, c\ell) = \sum_{\substack{(x,y)\in V_2,\\ \mathcal{C}(x)=\mathcal{C}(y)}} (1 - \mathbb{1}[\ell(x, y) = c\ell(\mathcal{C}(x))]) + \sum_{\substack{(x,y)\in V_2,\\ \mathcal{C}(x)\neq\mathcal{C}(y)}} (1 - \mathbb{1}[\ell(x, y) = l_0]), \quad (2)$$

where $\mathbb{1}[\cdot]$ denotes the indicator function. □

Equation (2) is composed of two terms, representing intra- and inter-cluster costs, respectively. In particular, the intra-cluster-cost term aims to measure the homogeneity of the labels on the intra-cluster edges: a pair of vertices (x, y) assigned to the same cluster pays a cost if and only if their relation type $\ell(x, y)$ is different from the predominant relation type of the cluster as indicated by the function $c\ell$. On the other hand, as far as the inter-cluster cost, the objective function penalizes two vertices x and y unless they are adjacent (i.e., unless $\ell(x, y) \neq l_0$); instead, if x and y are adjacent, the objective function incurs a cost, regardless of the label $\ell(x, y)$ on the shared edge.

Example 2.1. *Consider the problem instance in Figure 1(a), along with the solution in Figure 1(b), where the two clusters depicted are labeled with the* `green` *and* `blue` *label, respectively. The intra-cluster cost of the solution is equal to the number of intra-cluster edges that have a label different from the one assigned to the corresponding cluster. As both the clusters are mono-chromatic cliques, the resulting overall intra-cluster cost is zero. The overall cost of the solution is thus equal to the inter-cluster cost only, which corresponds to the number of edges between vertices in different clusters and is therefore equal to 5.* □

It is easy to observe that, when $|L| = 1$, the CHROMATIC-CORRELATION-CLUSTERING problem corresponds to the binary version of CORRELATION-CLUSTERING. Thus, our problem is a generalization of the standard problem. Since CORRELATION-CLUSTERING is **NP**-hard [Bansal et al. 2004], we can easily conclude that CHROMATIC-CORRELATION-CLUSTERING is **NP**-hard as well.

The previous observation motivates us to consider whether applying standard CORRELATION-CLUSTERING algorithms, just ignoring the different colors, would be a good solution to the problem. As we show in the following example, such an approach is not guaranteed to produce a good solution.

Example 2.2. *For the problem instance in Figure 1(a), the optimal solution of the standard* CORRELATION-CLUSTERING *problem that ignores edge colors, would be composed of a single cluster containing all the six vertices, as, according to Equation (1), this solution has*

a (minimum) cost of 4 corresponding to the number of missing edges within the cluster. Conversely, this solution has a non-optimal cost 12 when evaluated in terms of the CHROMATIC-CORRELATION-CLUSTERING *objective function, i.e., according to Equation (2). Instead, the optimum in this case corresponds to the cost-5 solution depicted in Figure 1(b).* □

2.2. Multi-chromatic correlation clustering

We now shift the attention to the case where object relations can be expressed by more than one label, i.e., the input to our problem is an edge-labeled graph whose edges may have multiple labels. To deal with this more general version of the problem, we extend the formulation in Problem 2 by (*i*) measuring the intra-cluster label homogeneity by means of a distance function d_ℓ between sets of labels, and (*ii*) allowing the output cluster label function $c\ell$ to assign *a set* of labels to each cluster (instead of a single label). The formal definition of our problem is as follows.[1]

PROBLEM 3 (MULTI-CHROMATIC-CORRELATION-CLUSTERING). *Let $G = (V, E, L, l_0, \ell)$ be an edge-labeled graph, where V is a set of vertices, $E \subseteq V_2$ is a set of edges, L is a set of labels, $l_0 \notin L$ is a* special *label, and $\ell : V_2 \to 2^L \cup \{l_0\}$ is a labeling function that assigns a set of labels to each unordered pair of vertices in V such that $\ell(x, y) = l_0$ if and only if $(x, y) \notin E$. Let also $d_\ell : 2^L \cup \{l_0\} \times 2^L \cup \{l_0\} \to \mathbb{R}^+$ be a distance function between sets of labels. We want to find a clustering $\mathcal{C} : V \to \mathbb{N}$ and a cluster labeling function $c\ell : \mathcal{C}[V] \to 2^L$ so as to minimize the cost*

$$\text{cost}(G, \mathcal{C}, c\ell) = \sum_{\substack{(x,y)\in V_2,\\ \mathcal{C}(x)=\mathcal{C}(y)}} d_\ell(\ell(x, y), c\ell(\mathcal{C}(x)) + \sum_{\substack{(x,y)\in V_2,\\ \mathcal{C}(x)\neq\mathcal{C}(y)}} d_\ell(\ell(x, y), \{l_0\}). \quad \square \tag{3}$$

It is easy to see that MULTI-CHROMATIC-CORRELATION-CLUSTERING is a generalization of CHROMATIC-CORRELATION-CLUSTERING. In particular, MULTI-CHROMATIC-CORRELATION-CLUSTERING reduces to CHROMATIC-CORRELATION-CLUSTERING when all edges in the input graph are single-labeled and the distance d_ℓ is defined as

$$d_\ell(\{l_1\}, \{l_2\}) = \begin{cases} 0, & \text{if } l_1 = l_2 \\ 1, & \text{otherwise.} \end{cases}$$

A key point of differentiation between the multiple-label and single-label formulations is that the former employs a distance function d_ℓ to measure how much label sets differ from each other. Among the various possible choices of d_ℓ, in this work we use the popular Hamming distance, as a good tradeoff between simplicity and effectiveness. The Hamming distance between two sets of labels $L_1 \subseteq L$, $L_2 \subseteq L$ is defined as the number of disagreements between L_1 and L_2:

$$d_\ell(L_1, L_2) = |L_1 \setminus L_2| + |L_2 \setminus L_1|.$$

In our context, thus, the distance d_ℓ lies within the range $[0..|L| + 1]$. Also, this choice of distance makes the cost paid by an inter-cluster edge in Equation (3) equal to the number of labels present on that edge (plus 1), i.e., $d_\ell(\ell(x, y), \{l_0\}) = |\ell(x, y)| + 1$, $\forall (x, y) \in E$. This is desirable, as we recall that in the ideal output clustering every pair of vertices in different clusters should have a relation represented by the l_0 label, which models the "no-edge" case. Thus, the cost of an inter-cluster edge should be a function of the size of the label set present on that edge: the larger the number of labels, the further that relation from the absent-edge case.

3. ALGORITHMS FOR CHROMATIC-CORRELATION-CLUSTERING

3.1. The Chromatic Balls algorithm

We present next a randomized approximation algorithm for the CHROMATIC-CORRELATION-CLUSTERING problem. This algorithm, called Chromatic Balls, is motivated by the Balls algorithm [Ailon et al. 2008], which is an approximation algorithm for standard CORRELATION-CLUSTERING.

For completeness, we briefly review the Balls algorithm. The algorithm takes as input an undirected graph and processes it in iterations. In each iteration, the algorithm produces a cluster, and the vertices participating in the cluster along with the edges incident to such

[1]For the sake of presentation, in Problem 3 and in the remainder of the paper, we assume that the powerset 2^L does not include the empty set, i.e., $2^L = \{S \subseteq L \mid |S| \geq 1\}$.

Algorithm 1 Chromatic Balls

Input: Edge-labeled graph $G = (V, E, L, l_0, \ell)$, where $\ell : V_2 \to L \cup \{l_0\}$
Output: Clustering $\mathcal{C} : V \to \mathbb{N}$; cluster labeling function $c\ell : \mathcal{C}[V] \to L$

```
i ← 1
while E ≠ ∅ do
  pick an edge (x, y) ∈ E uniformly at random
  C ← {x, y} ∪ {z ∈ V | ℓ(x, z) = ℓ(y, z) = ℓ(x, y)}
  C(x) ← i, for all x ∈ C
  cℓ(i) = ℓ(x, y)
  remove C from G: V ← V \ C, E ← E \ {(x, y) ∈ E | x ∈ C}
  i ← i + 1
end while
for all x ∈ V do
  C(x) ← i
  cℓ(i) ← a label from L
  i ← i + 1
end for
```

vertices are removed from the graph. In particular, the algorithm picks as a pivot a vertex uniformly at random, and forms a cluster consisting of the pivot itself along with all vertices that are still in the graph and are adjacent to the pivot.

The outline of our Chromatic Balls is reported as Algorithm 1. The main difference with the Balls algorithm is that the edge labels are taken into account in order to build label-homogeneous clusters around the pivots. To this end, the pivot chosen at each iteration of Chromatic Balls is an edge (x, y), rather than a single vertex. The pivot edge is used to build a cluster around it: beyond the vertices x and y, the cluster C being formed contains all other vertices z still in the graph for which the *triangle* (x, y, z) is monochromatic, that is, $\ell(x, y) = \ell(y, z) = \ell(x, y)$. Since the label $\ell(x, y)$ forms the basis for creating the cluster C, the cluster inherits this label. All vertices in C along with all their incident edges are removed from the graph, and the main cycle of the algorithm terminates when there are no edges anymore. Eventually, all remaining vertices (if any) are made singleton clusters.

Running time. We now focus on the time complexity of the Chromatic Balls algorithm. To this end, we denote by n and m the number of vertices and edges in the input graph, respectively.

The overall time complexity is determined by two main steps, i.e., (i) picking the pivot edges, and (ii) building the clusters around the pivots. To choose pivots uniformly at random in an efficient way, one can shuffle the whole edge set once, at the beginning of the algorithm, by employing a linear-time algorithm for generating a random permutation of a finite set of objects (e.g., the well-known *Knuth-shuffle* algorithm). Then, it suffices to follow the ordering determined by the shuffling algorithm and pick the first edge that is still in the graph. This way, each edge is accessed once, whether it is selected as pivot or not, and the phase of choosing pivots overall takes $\mathcal{O}(m)$ time.

Building a cluster C, instead, requires to access all neighbors of the pivot edge (x, y). As the current cluster is removed from the graph at the end of each iteration, the neighbors of a pivot are not considered again in the remainder of the algorithm. Thus, each edge in the graph is overall visited $\mathcal{O}(1)$ times, and the entire phase of building all the clusters takes $\mathcal{O}(m)$ time, which corresponds to the overall time complexity of the Chromatic Balls algorithm.

Theoretical analysis. We now analyze the quality of the solutions output by Chromatic Balls. Our main result, given in Theorem 3.3, shows that the approximation guarantee of the algorithm depends on the number of *bad triplets* incident on a pair of vertices in the input graph. The notion of bad triplet is defined below; however, we anticipate here that this corresponds to a constant-factor approximation guarantee for bounded-degree graphs.

We define a subset of three vertices $\{x, y, z\}$ to be a *bad triplet* (B-triplet) if the induced triangle has at least two edges and is non-monochromatic, i.e., $|\{(x, y), (x, z), (y, z)\} \cap E| \geq 2$ and $|\{\ell(x, y), \ell(x, z), \ell(y, z)\}| > 1$. We denote by $\mathcal{T}$ the set of all B-triplets for an instance of our problem. Moreover, given a pair $(x, y) \in V_2$, we let $\mathcal{T}_{xy} \subseteq \mathcal{T}$ denote the set of all B-triplets in $\mathcal{T}$ that contain both x and y, i.e., $\mathcal{T}_{xy} = \{t \in \mathcal{T} \mid x \in t, y \in t\}$, while $T_{max} = \max_{(x,y)\in V_2} |\mathcal{T}_{xy}|$ denotes the maximum number of B-triplets that contain a pair of vertices.

We consider an instance $G = (V, E, L, l_0, \ell)$ of Problem 2 and the output $\langle \mathcal{C}, c\ell \rangle$ of our Chromatic Balls algorithm on G, where, we recall, $\mathcal{C}$ is the output clustering while $c\ell$ is the cluster labeling function. We rewrite the cost function in Equation (2) as the sum of the costs paid by a single pair $(x, y) \in V_2$. To this end, in order to simplify the notation, we hereinafter write the cost by omitting $\mathcal{C}$ and $c\ell$ while keeping G only:

$$\text{cost}(G) = \sum_{(x,y)\in V_2} \text{cost}_{xy}(G), \tag{4}$$

where $\text{cost}_{xy}(G)$ denotes the aforementioned contribution of the pair (x, y) to the total cost.

A first result we exploit in our analysis is the following: a pair of vertices (x, y) pays a non-zero cost in a solution output by Chromatic Balls *only if* such a pair belongs to at least one B-triplet of the input graph. This result is formally stated in the following lemma.

LEMMA 3.1. *If* $\text{cost}_{xy}(G) > 0$, *then* $\mathcal{T}_{xy} \neq \emptyset$.

PROOF. According to the cost function defined in Equation (2), $\text{cost}_{xy}(G) > 0$ if and only if either 1) x and y are adjacent and the edge (x, y) is split (i.e., x and y are put in different clusters), or 2) x and y are placed in the same cluster C while $\ell(x, y)$ is not equal to the label of C. We analyze each case next.

1) According to the outline of Chromatic Balls, (x, y) is split when, at some iteration i, it happens that x is put into cluster C, while y is not, or vice versa. Assuming that the vertex chosen to belong to C is x (an analogous reasoning holds considering y as belonging to C), we have two subcases:
   (a) An edge (x, z) is chosen as pivot, with $z \neq y$. The fact that the edge (x, y) is split implies one of the following: either $\ell(x, y) \neq \ell(x, z)$ or $\ell(y, z) \neq \ell(x, z)$; each of the cases further implies that the triangle $\{x, y, z\}$ is a B-triplet.
   (b) The edge chosen as pivot is (z, w), with $\{z, w\} \cap \{x, y\} = \emptyset$. In this case, to have (i) (x, y) split and (ii) x put in the cluster being formed while y does not, it must be verified that either $\ell(z, y) \neq \ell(z, w)$ or $\ell(w, y) \neq \ell(z, w)$. Each of these cases implies that there exists a B-triplet containing x and y (i.e., either $\{x, y, z\}$ or $\{x, y, w\}$).
2) In this case, as both x and y are chosen as being part of the current cluster C, it must hold that the pivot chosen is (z, w), with $\{z, w\} \cap \{x, y\} = \emptyset$, and both $\{x, z, w\}$ and $\{x, y, w\}$ are mono-chromatic triangles. This fact, along with the hypothesis $\ell(x, y) \neq \ell(z, w)$, implies that both $\{x, y, z\}$ and $\{x, y, w\}$ are B-triplets.

The above reasoning shows that all situations where the pair (x, y) pays a non-zero cost imply the existence of at least one B-triplet that contains x and y. The lemma follows. □

A direct implication of Lemma 3.1 is that one can express the cost of the solution of our Chromatic Balls algorithm in terms of B-triplets only. Indeed, the lemma guarantees that vertex pairs that are not contained in a B-triplet pay no cost; thus, all non-B-triplets can safely be discarded. For every $t \in \mathcal{T}$, let α_t denote the cost paid by t in a Chromatic Balls solution. Note that α_t can be less than 3, because not all the three vertex pairs of t are charged against t, as a vertex pair may participate in more than one B-triplet of the input graph, but the pair's contribution to the overall cost is at most one. As a consequence, the $\text{cost}(G)$ can be expressed as:

$$\text{cost}(G) = \sum_{t\in\mathcal{T}} \alpha_t.$$

Let us now focus on the cost of an *optimal* solution for our problem instance G. Let $\text{cost}^*(G)$ denote such an optimal (i.e., minimum) cost. As shown in the next lemma, we can lower bound $\text{cost}^*(G)$ by using α_t.

LEMMA 3.2. *The cost of the optimal solution on graph G has the following bound*

$$\text{cost}^*(G) \;\geq\; \frac{1}{3T_{max}} \sum_{t\in\mathcal{T}} \alpha_t.$$

PROOF. We start by observing that a B-triplet incurs a non-zero cost in every solution, and thus in the optimal solution as well. Here by "a non-zero cost incurred by a B-triplet" we mean a non-zero cost paid by at least one of the vertex pairs in that B-triplet. Then, a lower bound on the cost of the optimal solution $\text{cost}^*(G)$ can be obtained by counting the number of disjoint B-triplets in the input. This lower bound can alternatively be expressed

by considering the whole set of (not necessarily edge-disjoint) B-triplets in the input by restating the following result by Ailon et al. [Ailon et al. 2008]: denoting by $\{\beta_t \mid t \in \mathcal{T}\}$ any assignment of nonnegative weights to the B-triplets in $\mathcal{T}$ satisfying $\sum_{t\in\mathcal{T}_{xy}} \beta_t \leq 1$ for all $(x, y) \in V_2$, it holds that $\text{cost}^*(G) \geq \sum_{t\in\mathcal{T}} \beta_t$.

Now, we note that, as $\alpha_t \leq 3$, then, for every vertex pair (x, y) it holds that $\sum_{t\in\mathcal{T}_{xy}} \alpha_t \leq 3|\mathcal{T}_{xy}| \leq 3T_{max}$; and thus $\sum_{t\in\mathcal{T}_{xy}} \frac{\alpha_t}{3T_{max}} \leq 1$, for all $(x, y) \in V_2$. This implies that setting $\beta_t = \frac{\alpha_t}{3T_{max}}$ suffices to satisfy the condition $\sum_{t\in\mathcal{T}_{xy}} \beta_t \leq 1$ for all $(x, y) \in V_2$, and thus the result by Ailon et al. applies:

$$\text{cost}^*(G) \;\geq\; \sum_{t\in\mathcal{T}} \beta_t \;=\; \frac{1}{3T_{max}} \sum_{t\in\mathcal{T}} \alpha_t.$$

□

The final approximation factor of the Chromatic Balls algorithm can now be easily derived by combining Lemma 3.1 and Lemma 3.2. Such a result is formally stated in the following theorem.

THEOREM 3.3. *The approximation ratio of the Chromatic Balls algorithm on input G is*

$$\frac{\text{cost}(G)}{\text{cost}^*(G)} \;\leq\; 3T_{max}.$$

PROOF. Immediate from Lemma 3.1 and Lemma 3.2:

$$\frac{\text{cost}(G)}{\text{cost}^*(G)} \;\leq\; \frac{\sum_{t\in\mathcal{T}} \alpha_t}{\frac{1}{3T_{max}} \sum_{t\in\mathcal{T}} \alpha_t} \;=\; 3T_{max}.$$

□

Theorem 3.3 shows that the approximation factor of the Chromatic Balls algorithm is bounded by the maximum number T_{max} of B-triplets that contain a specific pair of vertices. The result quantifies the quality of the performance of the algorithm as a property of the input graph. For example, as the following corollary shows, the algorithm provides a constant-factor approximation for bounded-degree graphs.

COROLLARY 3.4. *The approximation ratio of the Chromatic Balls algorithm on input G is*

$$\frac{\text{cost}(G)}{\text{cost}^*(G)} \leq 6\,(D_{max} - 1)\,,$$

where $D_{max} = \max_{x\in V} |\{(x, y) \mid (x, y) \in E\}|$ is the maximum degree of a vertex in G.

PROOF. By definition, any B-triplet must contain at least two adjacent vertices. Thus, the number of B-triplets containing a pair (x, y) is upper bounded by the number of neighbors of x plus the neighbors of y minus 2, which is clearly $\leq 2\ D_{max} - 2$. This leads to $3T_{max} \leq 6(D_{max} - 1)$, which proves the corollary. □

3.2. Enhancing the Chromatic Balls algorithm

In this section we present two additional algorithms for the CHROMATIC-CORRELATION-CLUSTERING problem. The first one is a variant of the Chromatic Balls algorithm that employs two heuristics in order to overcome some weaknesses of Chromatic Balls. The second one is an *alternating-minimization* method that is designed to directly optimize the objective function so as to reach a local optimum.

3.2.1. *Lazy Chromatic Balls.*

The algorithm we present next is motivated by the following example, in which we discuss what may go wrong during the execution of the Chromatic Balls algorithm.

Example 3.5. *Consider the graph in Figure 2: it has a fairly evident* `green` *cluster composed of all vertices in the graph but* S *and* T *(i.e, vertices* {U,V,R,X,Y,W,Z}*).*

However, as all the edges have the same probability of being selected as pivots, Chromatic Balls might miss this `green` *cluster, depending on which edge is selected first. For instance, suppose that the first pivot chosen is (Y,S). Chromatic Balls forms the* `red` *cluster* {Y,S,T} *and removes it from the graph. Removing vertex Y makes the edge (X,Y) missing, which would have been a good pivot to build a* `green` *cluster. At this point, even if the second*

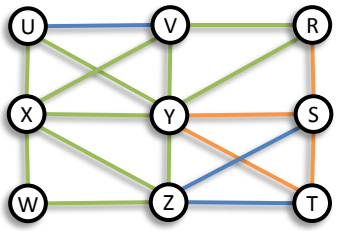


Fig. 2: An example of an edge-labeled graph.

Algorithm 2 Lazy Chromatic Balls

Input: Edge-labeled graph $G = (V, E, L, l_0, \ell)$, where $\ell : V_2 \to L \cup \{l_0\}$;
Output: Clustering $\mathcal{C} : V \to \mathbb{N}$; cluster labeling function $c\ell : \mathcal{C}[V] \to L$

$i \leftarrow 1$
while $E \neq \emptyset$ **do**
  pick a random vertex $x \in V$ with probability proportional to $\Delta(x)$
  pick a random vertex $y \in \{z \in V \mid (x, z) \in E\}$ with probability proportional to $\delta(y, \lambda(x))$
  $C \leftarrow \{x, y\} \cup \{z \in V \mid \ell(x, z) = \ell(y, z) = \ell(x, y)\}$
  repeat
    $C' \leftarrow \{z \in V \mid \exists w \in C \wedge \ell(x, z) = \ell(w, z) = \ell(x, w)\}$
    $C'' \leftarrow \{z \in V \mid \exists w \in C \wedge \ell(y, z) = \ell(w, z) = \ell(y, w)\}$
    $C \leftarrow C \cup C' \cup C''$
  until $C' \cup C'' = \emptyset$
  $\mathcal{C}(x) \leftarrow i$, for all $x \in C$
  $c\ell(i) = \ell(u, v)$
  remove C from G: $V \leftarrow V \setminus C$, $E \leftarrow E \setminus \{(x, y) \in E \mid x \in C\}$
  $i \leftarrow i + 1$
end while
for all $x \in V$ **do**
  $\mathcal{C}(x) \leftarrow i$
  $c\ell(i) \leftarrow$ a label from L
  $i \leftarrow i + 1$
end for

selected pivot edge is a `green` *one, say* (X,Z)*,* Chromatic Balls *would form only a small* `green` *cluster* {X,W,Z}. □

Motivated by the previous example we introduce the Lazy Chromatic Balls heuristic, which tries to minimize the risk of bad choices. Given a vertex $x \in V$, and a label $l \in L$, let $\delta(x, l)$ denote the number of edges incident on x having label l. Also, let $\Delta(x) = \max_{l \in L} \delta(x, l)$, and $\lambda(x) = \arg\max_{l \in L} \delta(x, l)$. Lazy Chromatic Balls differs from Chromatic Balls for two reasons:

Pivot random selection. At each iteration Lazy Chromatic Balls selects a pivot edge in two steps. First, a vertex x is picked up with probability directly proportional to $\Delta(x)$. Then, a second vertex y is selected among the neighbors of x with probability proportional to $\delta(y, \lambda(x))$.

Ball formation. Given the pivot (x, y), Chromatic Balls forms a cluster by adding all vertices z such that $\langle x, y, z\rangle$ is a monochromatic triangle. Lazy Chromatic Balls instead, iteratively adds vertices z in the cluster as long as they form a triangle $\langle X, Y, z\rangle$ of color $\ell(x, y)$, where X is either x or y, and Y can be any other vertex already belonging to the current cluster.

We remark that the heuristics at the basis of Lazy Chromatic Balls require no input parameters: the probabilities of picking a pivot are indeed defined based on $\Delta(\cdot)$ and $\delta(\cdot, \cdot)$, which are properties of the input graph. The algorithm is therefore parameter-free like Chromatic Balls. The details of Lazy Chromatic Balls are reported as Algorithm 2, while in the following example we further explain how the Lazy Chromatic Balls actually works.

Example 3.6. *Consider again Figure 2. Vertices* X *and* Y *have the maximum number of edges of one color: they both have 5* `green` *edges. Hence, one of them is chosen as the first pivot vertex* x *by* Lazy Chromatic Balls *with higher probability than the remaining vertices.*

Algorithm 3 Alternating Minimization

Input: Edge-labeled graph $G = (V, E, L, l_0, \ell)$, where $\ell : V_2 \to L \cup \{l_0\}$; number K of output clusters
Output: Clustering $\mathcal{C} : V \to \mathbb{N}$; cluster labeling function $c\ell : \mathcal{C}[V] \to L$
1: initialize $\mathcal{C}$ and $c\ell$ at random, such that $\mathcal{C}[V] = \{1, \ldots, K\}$
2: let $x_1, \ldots, x_n$ be the vertices in V ordered in some way
3: **repeat**
4: **for all** $k \in \{1, \ldots, K\}$ **do**
5: $c\ell(k) \leftarrow$ the majority label among the intra-cluster edges of cluster k
6: **end for**
7: **for** $i = 1, \ldots, n$ **do**
8: **if** cluster $\mathcal{C}(x_i)$ is not a singleton **then**
9: $k^* \leftarrow \arg\min_{k \in [1..K]} c_{ik}$ (Equation (5))
10: $\mathcal{C}(x_i) \leftarrow k^*$
11: **end if**
12: **end for**
13: **until** *convergence*

Suppose that X is picked up, i.e., $x = X$. Given this choice, the second pivot y is chosen among the neighbors of X with probability proportional to $\delta(y, \lambda(x))$, i.e., the higher the number of `green` *edges of the neighbor, the higher the probability for it to be chosen. In this case, hence,* Lazy Chromatic Balls *would likely choose Y as a second pivot vertex y, thus making (X,Y) the selected pivot edge. Afterwards,* Lazy Chromatic Balls *adds to the newly formed cluster the vertices {U,V,Z} because each of them forms a* `green` *triangle with the pivot edge. Then, R enters the cluster too, because it forms a* `green` *triangle with Y and V, which is already in the cluster. Similarly, W enters the cluster thanks to Z.* □

Running time. Picking the first pivot x can be implemented with a priority queue with priorities $\Delta \times rnd$, where rnd is a random number. This requires computing Δ for all vertices, which takes $\mathcal{O}(nh + m)$ time (where $h = |L|$). Putting all vertices in the queue takes $\mathcal{O}(n \log n)$ time.

The time complexity of the various steps of the main cycle (over all the iterations of the cycle) is as follows. Given x, the second pivot y is selected by probing all neighbors of x still in the graph (this is needed to pick up a neighbor of x with probability proportional to $\delta(y, \lambda(x))$. This takes $\mathcal{O}(m)$ time, as for each pivot x, its neighbors are accessed only once throughout the whole execution. Building the various clusters takes $\mathcal{O}(m)$ time as well, as it requires a traversal of the graph where each edge is accessed $\mathcal{O}(1)$ times. Finally, once a cluster C has been built, the priority with which the neighbors of each vertex in C are stored in the priority queue needs to be updated. This in turns requires updating (i) Δ for all of such vertices, which can be done in $\mathcal{O}(mh)$ time, and (ii) the priority queue based on the new Δ values computed, which takes $\mathcal{O}(m \log n)$ time.

In conclusion, therefore, the overall time complexity of the Lazy Chromatic Balls algorithm is $\mathcal{O}((h + \log n)\ m)$.

3.2.2. An alternating-minimization approach.

A nice feature of the previous algorithms is that they are parameter-free: they produce clusterings by using information that is local to the pivot edges, without forcing the number of output clusters in any way. However, in some cases, it would be desirable to have a pre-specified number K of clusters. To this end, we present here an algorithm based on the *alternating-minimization* paradigm [Csiszar and Tusnady 1984], that takes as input the number K of output clusters and attempts to minimize Equation (2) directly. The pseudocode of the proposed algorithm, called Alternating Minimization, is reported as Algorithm 3.

In a nutshell, Alternating Minimization starts with a random assignment of both vertices and labels to clusters (Line 1) and works by iteratively alternating between two optimization steps until convergence, i.e., until no changes, neither in $\mathcal{C}$ nor in $c\ell$, have been observed with respect to the previous iteration (Lines 3-13). In the first step, the algorithm finds the best label for every cluster given the current assignment of vertices to clusters (Lines 4-6). It is easy to see that an optimal solution of this step corresponds to simply assign to each cluster the label present on the majority of the edges connecting two vertices in that cluster (with ties broken arbitrarily). In the second step, the algorithm finds the best cluster assignment

Algorithm 4 Multi-Chromatic Balls

Input: Edge-labeled graph $G = (V, E, L, l_0, \ell)$, where $\ell : V_2 \to 2^L \cup \{l_0\}$
Output: Clustering $\mathcal{C} : V \to \mathbb{N}$; cluster labeling function $c\ell : \mathcal{C}[V] \to 2^L$
$i \leftarrow 1$
while $E \neq \emptyset$ **do**
pick an edge $(x, y) \in E$ uniformly at random
$C \leftarrow \{x, y\} \cup \{z \in V \mid d_\ell(\ell(x, y), \ell(x, z)) = d_\ell(\ell(x, y), \ell(y, z)) = 0\}$
$\mathcal{C}(x) \leftarrow i$, for all $x \in C$
$c\ell(i) = \ell(x, y)$
$V \leftarrow V \setminus C$, $E \leftarrow E \setminus \{(x, y) \in E \mid x \in C\}$ (remove C from G)
$i \leftarrow i + 1$
end while
for all $x \in V$ **do**
$\mathcal{C}(x) \leftarrow i$
$c\ell(i) \leftarrow$ a label set from 2^L
$i \leftarrow i + 1$
end for

for every $x \in V$ given the assignments of every other $y \in V$ and the current cluster labels (Lines 7-12). Particularly, while updating the cluster labels in the first step does not depend on the order with which the various clusters are processed, for this second step we need to establish an ordering among the vertices in V and update cluster assignments following that ordering. Given a vertex x_i, the best cluster k^* for x_i is the cluster resulting in the minimum clustering cost (computed according to Equation (2)) when vertex x_i is assigned to it. Looking at Equation (2), it is easy to see that the cost paid by a single vertex x_i when assigned to a cluster k is equal to the sum of two terms: (i) the number of vertices $y \neq x_i$ within cluster k such that the label $\ell(x_i, y)$ is other than the label assigned to k, and (ii) the number of edges connecting x_i with some vertex in a cluster other than k:

$$c_{ik} = \sum_{\substack{y \in V \setminus \{x_i\}, \\ \mathcal{C}(y) = k}} (1 - \mathbb{1}[\ell(x_i, y) = c\ell(k)]) + \sum_{\substack{y \in V \setminus \{x_i\}, \\ \mathcal{C}(y) \neq k}} (1 - \mathbb{1}[\ell(x_i, y) = l_0]). \tag{5}$$

Therefore, the best cluster for x_i is the one that minimizes the above cost, that is $k^* = \arg\min_{k \in [1..K]} c_{ik}$ (Line 9).

Both the steps of the Alternating Minimization algorithm are solved optimally. As a consequence the value of the CHROMATIC-CORRELATION-CLUSTERING objective function is guaranteed to be non-increasing in every iteration of the algorithm, until convergence. Finding the global optimum is obviously hard, but the algorithm is guaranteed to converge to a *local minimum*.

Running time. The running time of Alternating Minimization is determined by the complexity of the two main steps performed at each iteration of the algorithm, i.e., computing cluster labels and assigning vertices to clusters.

Computing cluster labels requires a traversal of the graph to determine, for each label $l \in L$ and for each cluster C, the number of intra-cluster edges in C having label l. This information is then exploited for computing the majority intra-cluster-edge label of each cluster. As a result, the first step overall takes $\mathcal{O}(m + Kh)$ time.

To implement efficiently the second step, i.e., assigning vertices to clusters, one can build a data structure where the cost c_{ik} of vertex x_i with respect to cluster k (computed according to Equation (5)) is accessed in constant time. It is easy to see that (re-)building such a data structure at every iteration of the algorithm requires just a traversal of the graph, thus taking $\mathcal{O}(m + Kn)$ time (the part $\mathcal{O}(Kn)$ is for initializing the data structure itself). This way, computing the optimal cluster k^* for all vertices takes $\mathcal{O}(Kn)$ time. Once the new cluster k^* of a vertex x_i has been computed, the data structure storing the costs c_{ik} needs to be updated. This can be done with a constant number of operations performed on every single neighbor of x_i; as a result, the global updating of the data structure considering all vertices $x_1, \ldots, x_n$ costs a traversal of the graph, i.e., $\mathcal{O}(m)$ time. Summing up all partial costs, the overall cost of the second main step of the algorithm turns out to be $\mathcal{O}(m + Kn)$.

In conclusion, as usually $h = |L| \ll n$, the time complexity of Alternating Minimization can be expressed as $\mathcal{O}(s(Kn + m))$, where s is the number of iterations to convergence.

4. ALGORITHMS FOR MULTI-CHROMATIC-CORRELATION-CLUSTERING

In this section we present an approximation algorithm for solving the MULTI-CHROMATIC-CORRELATION-CLUSTERING problem (Problem 3). The proposed algorithm, called Multi-Chromatic Balls, roughly follows the scheme of the Chromatic Balls algorithm designed for the single-label version of the problem.

The outline of our Multi-Chromatic Balls is reported as Algorithm 4. Like Chromatic Balls, the main idea of the algorithm is to pick a pivot edge (x, y) uniformly at random, build a cluster around it, and remove such a cluster from the graph. The process continues until there is no edge remaining to be picked as a pivot. A major peculiarity of Multi-Chromatic Balls is the presence of multiple labels on the edges of the input graph, which implies that the distance function d_ℓ between sets of labels should now be taken into account to determine how the cluster is built around the pivot (x, y). In particular, the cluster C defined at every iteration of the algorithm is composed of all vertices z such that the labels on the edges of the triangle (x, y, z) have all distance d_ℓ equal to zero, i.e., $d_\ell(\ell(x, y), \ell(x, z)) = d_\ell(\ell(x, y), \ell(y, z)) = 0$. According to our choice of distance d_ℓ (i.e., Hamming distance), the latter condition is equivalent to having label sets $\ell(x, y)$, $\ell(x, z)$, (y, z) equal to each other, i.e., $\ell(x, y) = \ell(x, z) = \ell(y, z)$. Finally, note that the presence of multiple labels on edges affects the way how the output clusters are labeled: following the common intuition, all clusters can now have more than one label assigned.

Running time. The time-complexity analysis of the Multi-Chromatic Balls algorithm is similar to Chromatic Balls. Choosing the pivots requires again $\mathcal{O}(m)$ time, while the step of selecting the vertices to be included in the current clusters requires visiting each edge in the graph at most once. But here, unlike Chromatic Balls, for each edge one needs to compute the distance d_ℓ to test whether the vertex being currently considered should be included into the current cluster. Assuming $\mathcal{O}(h)$ time to compute d_ℓ (which corresponds to the time, e.g., to compute the Hamming distance we consider in this work), the total time spent in the cluster-formation step is thus $\mathcal{O}(hm)$, which corresponds to the overall time complexity of the Multi-Chromatic Balls algorithm.

Theoretical analysis. The Multi-Chromatic Balls achieves a provable approximation guarantee, as formally stated in the following theorem.

THEOREM 4.1. *The approximation ratio of the Multi-Chromatic Balls algorithm on input* $G = (V, E, L, l_0, \ell)$ *is*

$$r(G) = \frac{\text{cost}(G)}{\text{cost}^*(G)} \leq 3hT_{max}.$$

PROOF. The MULTI-CHROMATIC-CORRELATION-CLUSTERING problem formulation can be reinterpreted from a single-label perspective by considering all labels in 2^L as a "single" label. The difference from the standard CHROMATIC-CORRELATION-CLUSTERING formulation is that the cost paid by a single vertex pair is this way bounded by $h = |L|$ (rather than being ≤ 1). Based on this observation, it is easy to see that running the Multi-Chromatic Balls algorithm on an instance G of the MULTI-CHROMATIC-CORRELATION-CLUSTERING problem is equivalent to running the Chromatic Balls algorithm on the same instance G reinterpreted from a single-label perspective. This means that a vertex pair (x, y) in the Multi-Chromatic Balls solution pays a non-zero cost only if the cost paid by the same vertex pair in the solution output by Chromatic Balls on the single-label reinterpretation of the problem instance is greater than zero. At the same time, however, the cost of (x, y) in the Multi-Chromatic Balls solution is bounded by h. Combining the latter two arguments implies that the approximation ratio of Multi-Chromatic Balls is guaranteed to be within a factor h of the approximation ratio achieved by the Chromatic Balls algorithm: the approximation ratio of Multi-Chromatic Balls is thus $3hT_{max}$. □

The above theorem states that the approximation ratio of Multi-Chromatic Balls increases the approximation ratio of Chromatic Balls by a factor $h = |L|$. However, this is not really problematic as the number of labels on real-world edge-labeled graphs is typically small and can safely be assumed to be constant. Moreover, as the following corollary shows, the approximation ratio remains constant for bounded-degree graphs.

COROLLARY 4.2. *The approximation ratio of the Multi-Chromatic Balls algorithm on input* G *is*

$$r(G) \leq 6h\,(D_{max} - 1)\,. \quad \square$$

Algorithm 5 Synthetic data generator

Input: number of vertices n, number of clusters K, number of labels h, probability p of intra-cluster edges, probability q of inter-cluster edges, probability w that an edge inside a cluster has a color different from the cluster
Output: edge labeled graph $G = (V, E, L, l_0, \ell)$

$V \leftarrow [1..n]$, $E \leftarrow \emptyset$, $L \leftarrow \{l_1, \ldots, l_h\}$
produce a clustering $\mathcal{C}$ by assigning each vertex $x \in V$ to a cluster selected uniformly at random
assign to each cluster a label selected uniformly at random from L
for all pairs $(x, y) \in V_2$ **do**
  pick 3 random numbers $r_1, r_2, r_3 \in [0, 1]$
  if $\mathcal{C}(x) = \mathcal{C}(y)$ **then**
    if $r_1 < p$ **then**
      if $r_2 < w$ **then**
        $E \leftarrow E \cup \{(x, y)\}$
        $\ell(x, y) \leftarrow$ a random label from $L \setminus \{c\ell(\mathcal{C}(x))\}$
      else
        $E \leftarrow E \cup \{(x, y)\}$
        $\ell(x, y) \leftarrow c\ell(\mathcal{C}(x))$
      end if
    end if
  else if $r_3 < q$ **then**
    $E \leftarrow E \cup \{(x, y)\}$
    $\ell(x, y) \leftarrow$ a random label from L
  end if
end for

5. EXPERIMENTAL EVALUATION

In this section we report experiments to test the validity of the proposed algorithms, i.e., Chromatic Balls, Lazy Chromatic Balls, Alternating Minimization, and Multi-Chromatic Balls, which, for the sake of brevity, we refer to as CB, LCB, AM, and M-CB, respectively. We also evaluate the performance of the baseline described in the Introduction, namely the "standard" Balls algorithm [Ailon et al. 2008], whichproduce a clustering $\mathcal{C}$ ignores colors. We refer to this baseline as B.

All algorithms are implemented in JAVA, and all experiments are performed on a single core of a desktop machine with Intel Core CPU at 2.90GHz and 16GB RAM. Also, as all algorithms are randomized, all measurements reported are averaged over 50 runs.

5.1. Experiments on synthetic data

We evaluate our (single-label) algorithms on synthetic datasets generated by the process outlined in Algorithm 5. In a nutshell, the generator initially assigns vertices and labels to clusters uniformly at random, and then adds noise according to the probability parameters p, q, and w. Given the assignment of vertices to clusters, intra-cluster edges are sampled with probability p, and they are given the correct label (the label of the cluster they are assigned to) with probability $1-w$, while, inter-cluster edges are sampled with probability q. Further details about the synthetic generator can be found in Appendix A.

The initial assignment of vertices and labels to clusters serves as a ground truth underlying the corresponding synthetic dataset. We compare the resulting clusterings with the ground-truth clustering using the well-known F-measure external cluster-validity criterion. Given a ground-truth clustering $\hat{\mathcal{C}}$ and a clustering solution $\mathcal{C}$ having $\hat{K}$ and K clusters, respectively, F-measure is defined in terms of *precision* and *recall* as follows:

$$F(\mathcal{C}, \widetilde{\mathcal{C}}) = \frac{1}{n} \sum_{\hat{k}=1}^{\hat{K}} S_{\hat{k}} \max_{k \in [1..K]} F_{\hat{k}k},$$

where $F_{\hat{k}k} = (2P_{\hat{k}k}R_{\hat{k}k})/(P_{\hat{k}k} + R_{\hat{k}k})$ such that $P_{\hat{k}k} = S_{\hat{k}\cap k}/S_k$ and $R_{\hat{k}k} = S_{\hat{k}\cap k}/S_{\hat{k}}$, while $S_{\hat{k}\cap k}$ denotes the number of common objects between cluster $\hat{k}$ of $\widetilde{\mathcal{C}}$ and cluster k of $\mathcal{C}$, and $S_{\hat{k}}$ and $S_{\hat{k}}$ are the sizes of clusters $\hat{k}$ and k, respectively. It easy to see that $F \in [0, 1]$.

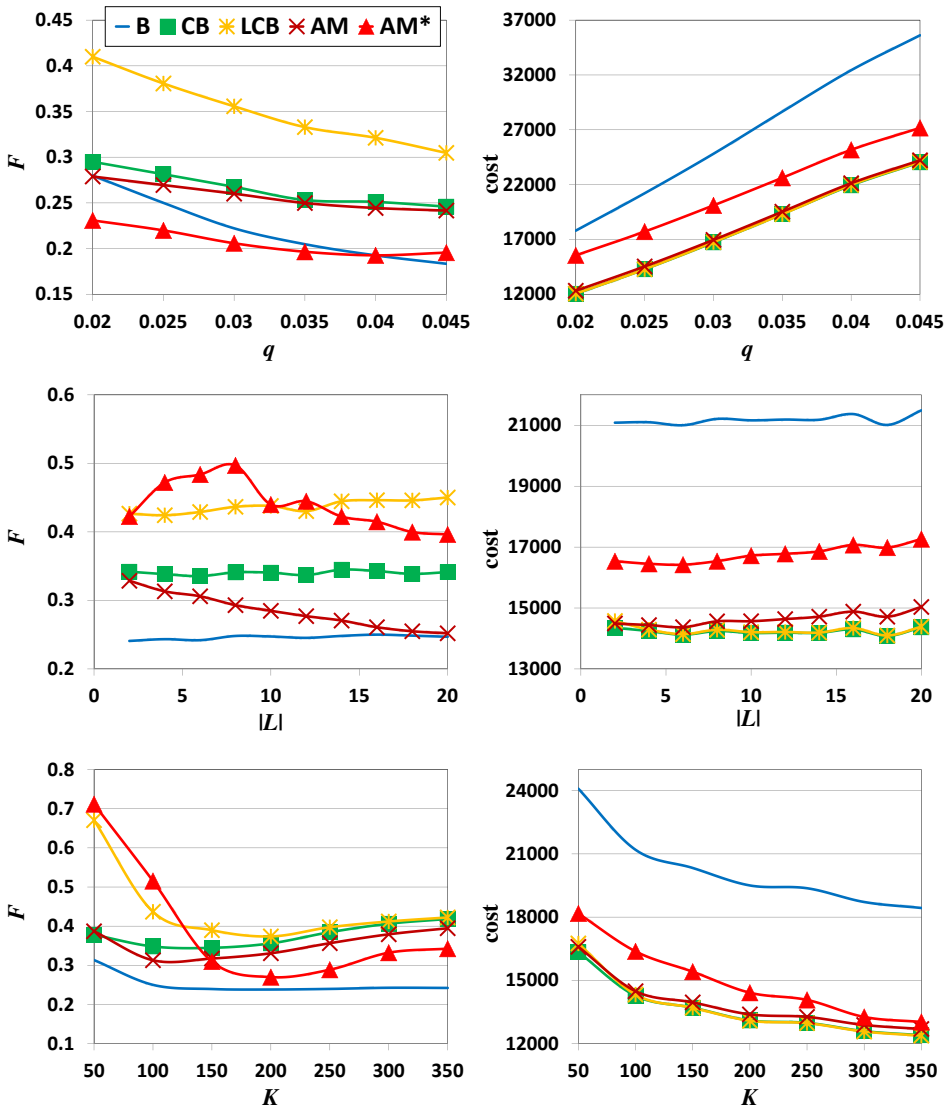


Fig. 3: Accuracy on synthetic datasets (single-label) in terms of F-measure (left) and solution cost (right), by varying level of noise (1st row), number of labels (2nd row), and number of ground-truth clusters (3rd row).

We generate datasets with a fixed number of vertices ($n = 1\,000$), and we vary (i) the noise level; (ii) the number of labels h; and (iii) the number of clusters K in the ground truth. To adjust the noise level, we vary one parameter among p, q, and w, while keeping fixed the other two; for the sake of brevity, we report results with varying q while keeping p and w set to 0.5.

As far as the number of clusters required as input by the AM algorithm, we consider two options: the average number of clusters output by the CB algorithm, and the number of clusters in the ground truth. We refer to these two settings by AM and AM*, respectively.

Results. In Figure 3 we report the performance of our algorithms in terms of F-measure, as well as solution cost (Equation (2)).

All trends observed by varying the parameters q, $h = |L|$, and K are intuitive. Indeed, for all methods, the performance decreases as the noise level q increases (Figure 3, 1st row). On the other hand, all methods give better solutions, in terms of cost, as the number of ground-truth clusters K increases (Figure 3, 3rd row, right). The reason is that, since CB and LCB tend to produce a large number of clusters, thus with a larger K the number of cluster discovered tends to become equal to the number of ground-truth clusters.

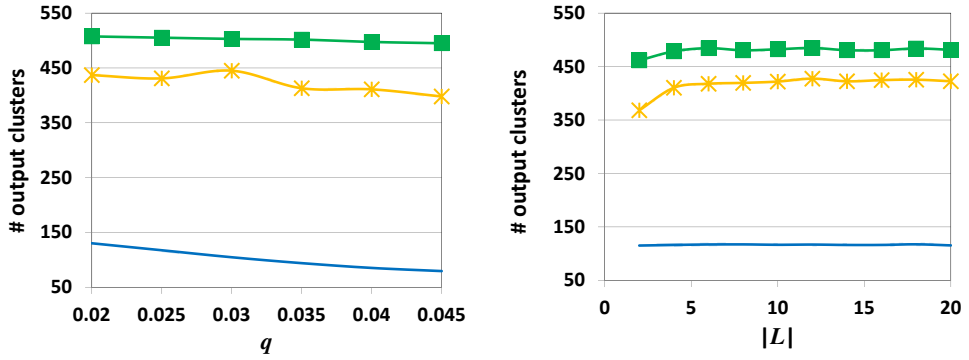


Fig. 4: Experiments on synthetic datasets: number of clusters by varying q (left) and $|L|$ (right).

Table I: Characteristics of real data. n: number of vertices; m: number of edges; D_{avg}: average degree; $|L|$: number of labels.

dataset	n	m	D_{avg}	$\|L\|$
String	18 152	401 582	44.25	4
Youtube	15 088	19 923 067	2 640.92	5
DBLP	312 416	2 110 470	13.51	100
Flickr	80 513	5 899 882	73.28	196

All proposed methods generally achieve both F-measure and solution cost results evidently better than the baseline. Particularly, in terms of solution cost, CB, LCB, and AM perform very close to each other and generally better than AM*. In terms of F-measure, instead, LCB is recognized as the best method in most cases.

In Figure 4 we report the number of clusters produced by B, CB, and LCB (we omit AM and AM* from the figure, as for these methods the number of clusters is pre-specified). The output clusters seem in general to be independent from the noise as well as the number of labels. However, as expected, we observe that CB tends to produce a larger number of clusters than LCB.

5.2. Experiments on real data

We experiment with four real datasets (Table I). The first three datasets (i.e., String, YouTube, and DBLP) represent edge-labeled graphs whose edges have only one label associated. The Flickr dataset is instead a graph where multiple labels are present on each edge.

String. A protein-protein interaction (PPI) network obtained from string-db.org, i.e., a database of known protein interactions for a large number of organisms. The dataset is an undirected graph where vertices represent proteins and edges represent protein interactions. Edges are labeled with 4 types of interactions, including direct (physical) and indirect (functional) associations.

Youtube. This dataset represents a network of associations in the `youtube` site. The vertices of the network represent users, videos, and channels. Entities in the network have five different types of associations: *contact*, *co-contact*, *co-subscription*, *co-subscribed*, and *favorite*; these are the edge labels considered by our algorithms. The dataset has been compiled by Tang et al. [Tang et al. 2009] and is available at http://socialcomputing.asu.edu/datasets/YouTube.

DBLP. This is a recent snapshot of the popular DBLP co-authorship network (http://dblp.uni-trier.de/xml/). For each co-authorship edge, we consider the bag of words obtained by merging the titles of all papers coauthored by the two authors. Words are stemmed and stop-words are removed. We then apply *Latent Dirichlet Allocation* (LDA) [Blei et al. 2003] to automatically identify 100 topics on each edge. After LDA topic-modeling, for each edge, we assign its most prominent topic discovered as an edge label.

Flickr. Flickr is a popular online community, where users share photos, participate in common-interest groups, and form friendships. The dataset is a publicly-available subset of the contact network underlying such a community containing information about friendship

Table II: Single-label results on real datasets: average cost

	cost			
dataset	B	CB	LCB	AM
String	163 305	160 060	155 881	156 976
Youtube	23 550 213	18 956 000	22 644 858	19 670 899
DBLP	2 260 065	1 633 149	1 678 714	2 018 952

Table III: Single-label results on real datasets: runtime (s)

	runtime (s)			
dataset	B	CB	LCB	AM
String	1.95	2.14	5.02	82.07
Youtube	5.89	6.78	16.15	273.36
DBLP	1.79	1.89	5.23	886.79

Table IV: Single-label results on real datasets: average number of output clusters

	#clusters			
dataset	B	CB	LCB	AM
String	1 086	1 451	784	1 451
Youtube	568	1 078	672	1 078
DBLP	66 276	123 197	99 948	123 197

between Flickr users and groups of interest joined by the users (http://socialcomputing.asu.edu/datasets/Flickr). An edge between two users exists if and only if they have the "friend" relationship. Every edge between two users is labeled with the IDs of all interest groups that are joined by both the users.

Single-label results. Tables II–IV summarize the results obtained on real data in terms of (average) solution cost, runtime (seconds), and number of output clusters, respectively. As already observed in synthetic data, all proposed algorithms clearly outperform the baseline B in terms of solution cost. CB is the best method on Youtube and DBLP, achieving up to 27.74% improvement with respect to the baseline in terms of solution cost. Instead, CB is slightly outperformed by LCB and AM on String, while LCB outperforms AM on String and DBLP.

As far as efficiency goes (Table III), we observe that all runtimes comply with the time-complexity analysis reported previously. Indeed, the baseline and the CB algorithm, which work both linearly in the number of edges of the input graph, are comparable to each other and outperform the other algorithms. LCB is slightly worse than CB (and B), while AM is in general the slowest method, mostly due to the multiple iterations needed for convergence. In general, however, all the proposed methods are very efficient, as they take a few seconds (CB and LCB) or minutes (AM), even on large and dense graphs like Youtube and DBLP.

Furthermore, Table IV shows that LCB tends to produce fewer clusters than CB, which conforms with the results observed in synthetic datasets and, in general, with the design principles of LCB.

Finally, as a qualitative example of the results produced by our methods, Figure 5 shows a cluster from the DBLP co-authorship network recognized by the LCB algorithm, containing 23 authors (vertices). Among the 71 intra-cluster edges, 58 have the same label, i.e., `Topic 18`, whose most representative (stemmed) keywords are: `queri, effici, spatial, tempor, search, index, similar, data, dimension, aggreg`. Other topics (edge colors) that appears are "sensor networks", "frequent pattern mining", "algorithms on graphs and trees", "support vector machines", "classifiers and Bayesian learning".

Multiple-label results. Let us now shift the attention to the multiple-label case. Table V summarizes the results obtained by our M-CB algorithm on the multiple-label Flickr dataset in term of solution cost (i.e., objective function value), runtime (seconds), and average number of clusters output. In the table we also report the corresponding results achieved by the baseline B.

As previously observed in the single-label case, the proposed M-CB consistently outperforms the baseline B in terms of accuracy. Particularly, M-CB reduces the cost of the baseline by roughly 27%. Concerning running times, the time complexity of M-CB increases the complexity of its single-label counterpart (i.e., CB) by a factor $h = |L|$. Thus, unlike the

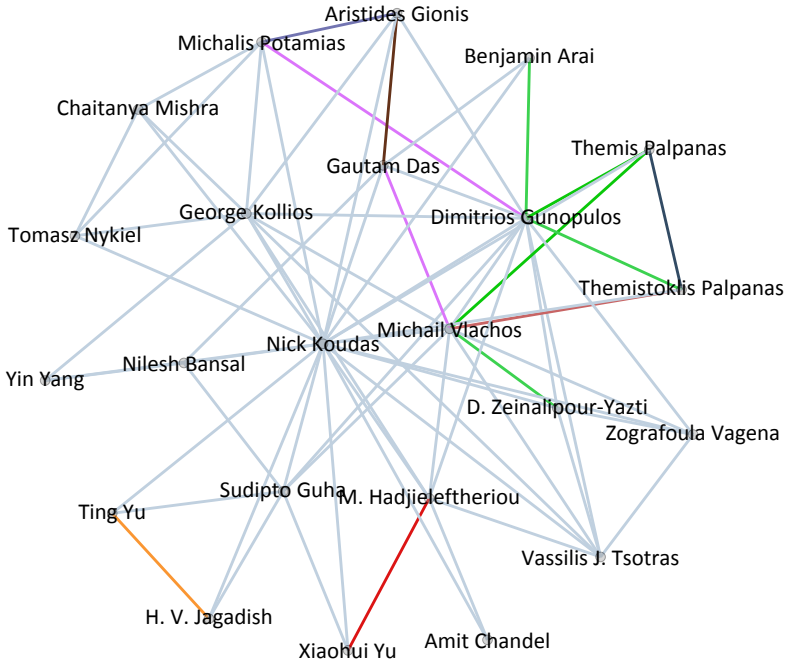


Fig. 5: An example cluster from DBLP.

Table V: Multiple-label results on the Flickr real dataset: cost, runtime (s), and average number of output clusters.

	B	M-CB
cost	8 487 181	6 222 204
runtime (s)	0.68	1.95
# clusters	27 032	41 814

single-label case, here we observe a gap between the runtimes of M-CB and the baseline. Nevertheless, such a gap remains quite small: the running time of M-CB is still within a few seconds.

6. RELATED WORK

The problem we study in this paper is a novel clustering problem that has not been explicitly studied before. Next we briefly overview related research on some neighborhood areas.

Edge-labeled graphs Graphs having edges labeled with a type of relation between the incident vertices are receiving increasing attention. So far, problems other than clustering have been studied on this kind of graph. Existing literature has mainly focused on reachability/shortest-path queries. [Jin et al. 2010] study the *subset-constrained reachability query* problem in edge-labeled graphs, which consists of checking whether two vertices are connected by a path having edge labels exclusively from a given set. [Xu et al. 2011] propose a partition-based algorithm for efficiently processing subset-constrained reachability queries, while [Fan et al. 2011] study reachability queries and graph pattern queries in graphs where both edges and vertices are labeled. [Rice and Tsotras 2010], instead, show that road networks have a natural representation in terms of edge-labeled graphs and study the problem of label-constrained shortest-path queries on that kind of network. The same problem is also studied in [Likhyani and Bedathur 2013; Bonchi et al. 2014].

Multidimensional networks. *Multidimensional* (or *multi-layered*) networks are networks defined as a set of graphs spanning the same set of entities. An intuitive representation of these networks is as edge-labeled graphs with multiple labels on the edges. Existing literature on multidimensional networks has focused on general characterization/analysis [Magnani and Rossi 2011; Berlingerio et al. 2011b; Kazienko et al. 2011], shortest paths [Brodka et al. 2011], and clustering [Tang et al. 2011; Berlingerio et al. 2011a; Rocklin and Pinar 2011]. The semantics of clustering this kind of network is however quite different from the problem we address in this work. Indeed, the goal behind clustering multidimensional networks is to

find a partitioning of vertices which is relevant in *all the dimensions (colors) at the same time*. In other words, in that setting each group of vertices is recognized as a good cluster only if it is actually good in the `green` network *and* the `red` network, and so on. In our work, we are rather interested in finding groups of objects that induce color-homogeneous clusters, which are not necessarily forced to be good for all colors simultaneously.

Close in spirit to clustering multidimensional networks, [Boden et al. 2012] propose *mining coherent subgraphs on multi-layered networks*, that is finding, for every possible subset of labels $L' \in 2^L$, all subgraphs that look *coherent* (i.e., densely connected and composed of edges with similar labels) with respect to L'. The problem studied by Boden et al. differs from the problem we tackle in this work as we are rather interested in finding a (single) partition of the vertices in the graph while *simultaneously* identifying the label (or the set of labels) that represent the various clusters well. In a sense, the work by Boden et al. is close to the classic problem of *subspace clustering* [Parsons et al. 2004], while our work can be rather interpreted as a reformulation of *projected clustering* [Kriegel et al. 2009] on edge-labeled graphs.

Correlation Clustering. The problem of CORRELATION-CLUSTERING was first defined by [Bansal et al. 2004] in its binary version. [Ailon et al. 2008] proposed the Balls algorithm that achieves expected approximation factor 5 if the weights obey the probability condition. If the weights obey the triangle inequality too, then the algorithm achieves a 2 approximation guarantee. Giotis and Guruswami [Giotis and Guruswami 2006] consider correlation clustering when the number of clusters is fixed. [Ailon and Liberty 2009] study a variant of correlation clustering where the goal is to minimize the number of disagreements between the produced clustering and a given ground truth clustering, while [Ailon et al. 2012] study the problem of bipartite correlation clustering. Finally, the correlation-clustering formulation has been recently extended to allow overlapping clusters [Bonchi et al. 2013].

We remark that none of the existing correlation-clustering variants considers edge-labeled graphs.

7. CONCLUSION AND FUTURE WORK

In this paper we introduce a novel clustering problem where the pairwise relations between objects are categorical. The problem has interesting applications, such as clustering social networks where individuals share different types of relations, or clustering protein networks, where proteins are associated with different types of interactions. We propose four algorithms and evaluate them on several synthetic and real datasets.

Our problem is a novel clustering formulation well-suited for mining, e.g., multi-labeled and heterogeneous datasets that are becoming increasingly common. With respect to the earlier version presented in [Bonchi et al. 2012], a major extension provided here is the capability of handling datasets where relationships between objects are described by multiple labels. However, we believe that there are still many other interesting extensions and fruitful future research directions. Among others, we would like to extend the problem formulation in order to output *overlapping clusters*, as well as investigate how the theoretical results about our synthetic generator discussed in Appendix A can be exploited to design further approximation algorithms. Other interesting directions include the use of other distance measures (beyond the Hamming distance) in the MULTI-CHROMATIC-CORRELATION-CLUSTERING problem, the extension of the LCB and AM algorithms to the multiple-label context, and the definition of cluster validity criteria that are specific of the context of edge-labeled graphs.

APPENDIX

A. SYNTHETIC RANDOM GENERATOR: RECOVERING THE HIDDEN GROUND TRUTH

We prove next how to recover with high probability the ground truth of a dataset randomly generated according to the synthetic generator described in Section 5. This result can be particularly useful for designing further approximation algorithms based on the implicit assumption that the dataset observed has been generated according to the synthetic generator at hand.

To prove the desired result, we use the classical result by McSherry [McSherry 2001] and the following Chernoff-type bound.

THEOREM A.1 ([CHERNOFF 1981]). *Let $X_1, X_2, \ldots, X_k$ be independently distributed $\{0,1\}$ variables with $E[X_i] = p$. Then, for every $t > 0$, we have*

$$Pr\left[\left|\sum_{i=1}^{k} X_i - \mathbb{E}[X]\right| > t\right] \leq 2e^{-2t^2/k}.$$

THEOREM A.2. *Let p, q, w the real numbers taken as input by the generator in Algorithm 5. There exists a constant c such that, if $w \neq \frac{l-1}{l}$ and $\frac{p-q}{p} > c\sqrt{\frac{\log n}{pn}}$, then, for sufficiently large n, we can recover $\langle \mathcal{C}, c\ell \rangle$ with high probability.*

PROOF SKETCH. Let S_i be the cardinality of the i-th cluster V_i, $i = 1, \ldots, k$. Also, define X_{ij} to be an indicator random variable such that $X_{ij} = 1$ if and only if vertex j goes to cluster V_i, $i = 1, \ldots, k, j = 1, \ldots, n$. Clearly, $S_i = \sum_{j=1}^{n} X_{ij}$ and $\mathbf{Pr}[X_{ij} = 1] = \frac{1}{k}$ and $\mathbb{E}[S_i] = \frac{n}{k}$. Let $d > 0$ be a constant. Applying Theorem A.1 by setting $t = \sqrt{\frac{d}{2} n \log n}$ where $d > 0$, and a union bound we obtain

$$\mathbf{Pr}\left[\exists \text{cluster } V_i \text{ s.t. } |V_i - \mathbb{E}[V_i]| \geq \sqrt{\frac{d}{2} n \log n}\right] \leq \frac{2k}{n^d} = o(1).$$

Hence with high probability $|V_i| = (1 + o(1))\frac{n}{k}$ for all $i = 1, \ldots, k$. We condition on this event, call it $\mathcal{G}$, and apply the Chernoff bound again. Specifically, let Y_i be the number of edges induced by cluster V_i. Then, $\mathbb{E}[Y_i] = p\binom{n/k}{2}$ for all $i = 1, .., k$.

$$\mathbf{Pr}\left[\exists \text{cluster } V_i \text{ s.t. } |Y_i - \mathbb{E}[Y_i]| \geq \frac{n}{2k}\sqrt{d \log n} \mid \mathcal{G}\right] \leq \frac{2k}{n^d} = o(1).$$

By removing the conditioning on event $\mathcal{G}$, we obtain that with high probability $Y_i = (1 + o(1))p\binom{n/k}{2}$.[2] In a similar way, the number of edges of cluster V_i with color $c\ell(V_i)$ is with high probability $(1 + o(1))p(1 - w)\binom{n/k}{2}$. For every other color than $c\ell(V_i)$ the number of edges with that color is $(1 + o(1))p\frac{w}{l-1}\binom{n/k}{2}$. This suggests the following two-step procedure for recovering $\mathcal{C}, c\ell$: (a) Apply Corollary 1 from [McSherry 2001] by setting $\delta = n^{-d}$ for some constant $d > 0$. This results in obtaining $\mathcal{C}$ with probability $1 - n^{-d}$. (b) Using the above concentration results, given that the condition $w \neq \frac{l-1}{l}$ guarantees that $p(1-w)\binom{n/k}{2} \neq p\frac{w}{l-1}\binom{n/k}{2}$, obtain $c\ell$. Hence, we can recover $\langle c\ell, \mathcal{C} \rangle$ with probability $1 - o(1)$ where the neglibible term goes down to 0 polynomially fast. □

REFERENCES

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

Nir Ailon, Moses Charikar, and Alantha Newman. 2008. Aggregating inconsistent information: Ranking and clustering. *Journal of the ACM (JACM)* 55 (2008), 23:1–23:27. Issue 5.

Nir Ailon and Edo Liberty. 2009. Correlation Clustering Revisited: The "True" Cost of Error Minimization Problems. In *Proc. Int. Colloquium on Automata, Languages and Programming (ICALP)*. 24–36.

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

M. Berlingerio, M. Coscia, and F. Giannotti. 2011a. Finding and Characterizing Communities in Multidimensional Networks. In *Proc. IEEE/ACM Int. Conf. on Advances in Social Networks Analysis and Mining (ASONAM)*. 490–494.

Michele Berlingerio, Michele Coscia, Fosca Giannotti, Anna Monreale, and Dino Pedreschi. 2011b. Foundations of Multidimensional Network Analysis. In *Proc. IEEE/ACM Int. Conf. on Advances in Social Networks Analysis and Mining (ASONAM)*. 485–489.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research (JMLR)* 3 (2003), 993–1022.

Brigitte Boden, Stephan Günnemann, Holger Hoffmann, and Thomas Seidl. 2012. Mining coherent subgraphs in multi-layer graphs with edge labels. In *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*. 1258–1266.

Francesco Bonchi, Aristides Gionis, Francesco Gullo, and Antti Ukkonen. 2012. Chromatic Correlation Clustering. In *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*. 1321–1329.

[2] The fact $\mathbf{Pr}[A] = \mathbf{Pr}[A|B]\mathbf{Pr}[B] + \mathbf{Pr}[A|\bar{B}]\mathbf{Pr}[\bar{B}]$ is exploited here to remove the conditioning.

Francesco Bonchi, Aristides Gionis, Francesco Gullo, and Antti Ukkonen. 2014. Distance oracles in edge-labeled graphs. In *Proc. Int. Conf. on Extending Database Technology (EDBT)*. 547–558.

Francesco Bonchi, Aristides Gionis, and Antti Ukkonen. 2013. Overlapping correlation clustering. *Knowledge and Information Systems (KAIS)* 35, 1 (2013), 1–32.

Piotr Brodka, Pawel Stawiak, and Przemyslaw Kazienko. 2011. Shortest Path Discovery in the Multi-layered Social Network. In *Proc. IEEE/ACM Int. Conf. on Advances in Social Networks Analysis and Mining (ASONAM)*. 497–501.

H. Chernoff. 1981. A Note on an Inequality Involving the Normal Distribution. *Annals of Probability* 9, 3 (1981), 533–535.

I. Csiszar and G. Tusnady. 1984. Information Geometry and Alternating Minimization Procedures. *Statistics and Decisions* (1984).

W. Fan, J. Li, S. Ma, N. Tang, and Y. Wu. 2011. Adding Regular Expressions to Graph Reachability and Pattern Queries. In *Proc. IEEE Int. Conf. on Data Engineering (ICDE)*. 39–50.

Ioannis Giotis and Venkatesan Guruswami. 2006. Correlation clustering with a fixed number of clusters. In *Proc. ACM-SIAM Symp. on Discrete Algorithms (SODA)*. 1167–1176.

R. Jin, H. Hong, H. Wang, N. Ruan, and Y. Xiang. 2010. Computing Label-Constraint Reachability in Graph Databases. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*. 123–134.

Przemyslaw Kazienko, Katarzyna Musial, Elżbieta Kukla, Tomasz Kajdanowicz, and Piotr Bródka. 2011. Multidimensional social network: model and analysis. In *Proc. Int. Conf. on Computational collective intelligence: technologies and applications (ICCCI)*. 378–387.

Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. 2009. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Trans. on Knowledge Discovery from Data (TKDD)* 3, 1 (2009), 1:1–1:58.

Ankita Likhyani and Srikanta Bedathur. 2013. Label constrained shortest path estimation. In *Proc. ACM Int. Conf. on Information & Knowledge Management (CIKM)*. 1177–1180.

Chuan Lin, Young rae Cho, Woo chang Hwang, Pengjun Pei, and Aidong Zhang. 2007. Clustering Methods in Protein-Protein Interaction Networks. In *Knowledge Discovery in Bioinformatics: Techniques, Methods and Application*, Xianhua Hu and Yi Pan (Eds.). Wiley.

Matteo Magnani and Luca Rossi. 2011. The ML-Model for Multi-layer Social Networks. In *Proc. IEEE/ACM Int. Conf. on Advances in Social Networks Analysis and Mining (ASONAM)*. 5–12.

F. McSherry. 2001. Spectral Partitioning of Random Graphs. In *Proc. IEEE Symp. on Foundations of Computer Science (FOCS)*. 529–537.

Lance Parsons, Ehtesham Haque, and Huan Liu. 2004. Subspace clustering for high dimensional data: a review. *SIGKDD Explorations Newsletter* 6, 1 (2004), 90–105.

M. Rice and V. J. Tsotras. 2010. Graph Indexing of Road Networks for Shortest Path Queries with Label Restrictions. *Proceedings of the VLDB Endowment (PVLDB)* 4 (2010), 69–80. Issue 2.

M. Rocklin and A. Pinar. 2011. On Clustering on Graphs with Multiple Edge Types. In *Proc. Workshop on Algorithms and Models for the Web Graph (WAW)*. 38–49.

Lei Tang, Xufei Wang, and Huan Liu. 2009. Uncovering Groups via Heterogeneous Interaction Analysis. In *Proc. IEEE Int. Conf. on Data Mining (ICDM)*. 503–512.

L. Tang, X. Wang, and H. Liu. 2011. Community detection via heterogeneous interaction analysis. *Data Mining and Knowledge Discovery (DAMI)* (2011), 1–33.

K. Xu, L. Zou, J. X. Yu, L. Chen, Y. Xiao, and D. Zhao. 2011. Answering Label-Constraint Reachability in Large Graphs. In *Proc. ACM Int. Conf. on Information and Knowledge Management (CIKM)*. 1595–1600.