Chromatic Correlation Clustering

Francesco Bonchi Aristides Gionis Francesco Gullo Antti Ukkonen
Yahoo! Research – Barcelona, Spain
{bonchi,gionis,gullo,aukkonen}@yahoo-inc.com

ABSTRACT

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 aims at partitioning the graph such that 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 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 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 via a strategy based on the *alternating minimization* paradigm.

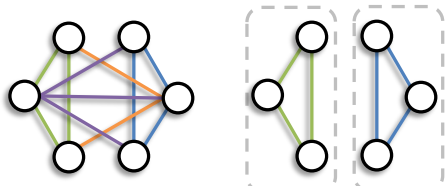
We test our algorithms on synthetic and real data from the domains of protein-interaction networks, social media, and bibliometrics. Experimental evidence show that our algorithms outperform a baseline algorithm both in the task of reconstructing a ground-truth clustering and in terms of objective function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

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Applications - *Data Mining*
Keywords: Clustering, Edge-labeled graphs.

1. INTRODUCTION

Clustering is one of the most 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 function $\text{sim}(x, y)$ representing the similarity between pairs of objects x and y. The similarity function is either provided explicitly as input, or it can be computed implicitly from the representation of the objects.

KDD'12, August 12–16, 2012, Beijing, China.



(a) (b)

Figure 1: An example of chromatic clustering: (a) input graph, (b) the optimal solution for chromatic-correlation-clustering (Problem 2).

In this paper, we consider a different clustering setting where the relationship among objects is represented by a relation type, such as a label $\ell(x, y)$ from a finite set of possible labels L. In other words, the *range* of the similarity function $\text{sim}(x, y)$ can be viewed as being *categorical*, instead of *numerical*. Moreover, we model the case where two objects x and y do not have any relation with a special label $l_0 \notin L$. Our framework has a natural graph interpretation: the input can be viewed as an edge-labeled graph $G = (V, E, L, \ell)$, where the set of vertices V is the set of objects to be clustered, the set of edges $E \subseteq V \times V$ is implicitly defined as $E = \{(x, y) \in V \times V \mid \ell(x, y) \neq l_0\}$, and each edge has a *label* in L or, as we like to think about it, a *color*.

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 (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 `green` edges. Aggregating such inconsistent information is resolved by optimizing a properly-defined objective function.

Applications. The study of edge-labeled graphs is motivated by many real-world applications and is receiving increasing attention in the data-mining literature [8, 10, 16]. As an 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 inter-

actions 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 [11].

As a further example, social networks are commonly represented as graphs, where the vertices represent individuals and the edges capture relationships among these individuals. Again, these relationships can be of various types, e.g., colleagues, neighbors, schoolmates, football-mates.

In bibliographic data, co-authorship networks represent collaborations among authors: in this case the topic of the collaboration can be seen as an edge label, and a cluster of vertices represents a topic-coherent community of researchers. In our experiments in Section 5 we show how our framework can be applied in all the above domains.

Contributions. In this paper we address the problem of clustering data with categorical similarity, achieving the following contributions:

- We define CHROMATIC-CORRELATION-CLUSTERING, a novel clustering problem for objects with categorical similarity, by revisiting the well-studied *correlation clustering* framework [3]. We show that our problem is a generalization of the traditional CORRELATION-CLUSTERING problem, implying that it is **NP**-hard.
- We introduce a randomized algorithm, named Chromatic Balls, that provides approximation guarantee proportional to the maximum degree of the graph.
- Though of theoretical interest, Chromatic Balls has some limits when it comes to practice. Trying to overcome these limits, 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 empirically assess our algorithms both on synthetic and real datasets. Experiments on synthetic data show that our algorithms outperform a baseline algorithm in the task of reconstructing a ground-truth clustering. Experiments on real-world data confirm that CHROMATIC-CORRELATION-CLUSTERING provides meaningful clusters.

The rest of the paper is organized as follows. In the next section we recall the traditional correlation clustering problem and introduce our new formulation. In Section 3 we introduce the Chromatic Balls algorithm and we prove its approximation guarantees. In Section 4 we present the two more practical algorithms, namely Lazy Chromatic Balls and Alternating Minimization. In Section 5 we report our experimental analysis. In Section 6 we discuss related work.

2. PROBLEM DEFINITION

Given a set of objects V, a clustering problem asks to partition the set V into clusters of similar objects. Assuming that cluster identifiers are represented by natural numbers, a clustering $\mathcal{C}$ can be seen 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 Numerous formulations and objective functions have been considered in the literature. One of these, considered both in the area of theoretical computer science and data mining, is that at the basis of the CORRELATION-CLUSTERING problem [3].

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

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

The intuition underlying the above problem is that the cost of assigning two objects x and y to the same cluster should be equal to the dissimilarity $1-\mathrm{sim}(x,y)$, while the cost of assigning the objects in different clusters should correspond to their similarity $\mathrm{sim}(x,y)$. A common case is when the similarity is binary, that is, $\mathrm{sim} : V \times V \to \{0,1\}$. In this case, Equation (1) reduces to counting the number of 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. Or equivalently, in a graph-based terminology, the objective function counts the number of "positive" edges that are cut plus the number of "negative" (i.e., non-existing) edges that are not cut.

In CHROMATIC-CORRELATION-CLUSTERING, which we formally define below, we still have negative edges (i.e., l_0-edges), but the positive edges may have different colors, representing different kinds of relations among the objects.

Problem 2 (CHROMATIC-CORRELATION-CLUSTERING)
Given a set V of objects, a set L of labels, a special label l_0, and a pairwise labeling function $\ell : V \times V \to L \cup \{l_0\}$, find a clustering $\mathcal{C} : V \to \mathbb{N}$ and a cluster labeling function $c\ell : \mathcal{C}[V] \to L$ so to minimize the cost

$$\mathrm{cost}(\mathcal{C}, c\ell) = \sum_{\substack{(x,y)\in V\times V, \\ \mathcal{C}(x)=\mathcal{C}(y)}} (1-\mathrm{I}[\ell(x,y) = c\ell(\mathcal{C}(x))]) + \sum_{\substack{(x,y)\in V\times V, \\ \mathcal{C}(x)\neq\mathcal{C}(y)}} \mathrm{I}[\ell(x,y)\neq l_0]. \quad (2)$$

Equation (2) is composed by two terms, representing intra- and inter-cluster costs, respectively. In particular, according to the intra-cluster cost term, any pair of objects (x,y) assigned to the same cluster should pay a cost if and only if their relation type $\ell(x,y)$ is other than the predominant relation type of the cluster indicated by the function $c\ell$. For the inter-cluster cost, the objective function does not penalize a pair of objects (x,y) only if they do not have any relation, i.e., $\ell(x,y) = l_0$. If $\ell(x,y) \neq l_0$, the objective function incurs a cost, regardless of the label $\ell(x,y)$.

Example 1 *For the problem instance in Figure 1(a), the solution in Figure 1(b) has a cost of 5: there is no intra-cluster cost, because the two clusters are cliques and their edges are monochromatic, while we have an inter-cluster cost of 5 as equal to the number of edges that are cut.*

It is trivial 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, we can easily conclude that CHROMATIC-CORRELATION-CLUSTERING is **NP**-hard too.

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

Example 2 *For the problem instance in Figure 1(a), the optimal solution for the standard* CORRELATION-CLUSTERING *which does not consider the different colors, would be composed by 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 according to the* CHROMATIC-CORRELATION-CLUSTERING *formulation, i.e., according to Equation (2). Instead, the optimum in this case would correspond to the cost 5 solution depicted in Figure 1(b).*

Although the example shows that for the chromatic version of the problem we cannot directly apply algorithms developed for the CORRELATION-CLUSTERING problem, we can use such algorithms at least as a starting point, as shown in the next section.

3. 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 [1], which is an approximation algorithm for standard CORRELATION-CLUSTERING.

For completeness, we briefly review the Balls algorithm. The algorithm works in iterations. Initially all objects are considered *uncovered*. In each iteration the algorithm produces a cluster, and the objects participating in the cluster are considered *covered*. In particular, the algorithm picks as pivot a random object currently uncovered, and forms a cluster consisting of the pivot itself along with all currently uncovered objects that are connected to the pivot.

The outline of our Chromatic Balls is summarized in Algorithm 1. The main difference with the Balls algorithm is that the edge labels are taken into account in order to build clusters around the pivots. To this end, the pivot chosen at each iteration of Chromatic Balls is an edge, thus a pair of objects, rather than a single object. The Chromatic Balls algorithm employs a set V' to keep all the objects that have not been assigned to any cluster yet; hence, initially, $V' = V$. At each iteration, a random edge (u, v) such that both objects u and v are currently in the set V' is selected as pivot (line 3). Given the pivot (u, v), a cluster C is formed around it. Beyond the objects u and v, the cluster C additionally contains all other objects $x \in V'$ for which the *triangle* (u, v, x) is monochromatic, that is, $\ell(u, x) = \ell(v, x) = \ell(u, v)$ (lines 4 and 5). Since the label $\ell(u, v)$ forms the basis for creating the cluster C, the cluster is labeled with this label (line 6). All objects added in C are removed from V' (line 7), and the algorithm terminates when V' does not contain any pair of objects that share an edge, i.e., that is labeled with a label other than l_0 (line 2). All objects remaining in the set V', if any, are eventually made singleton clusters (lines 8-11).

Computational complexity. The complexity of the Chromatic Balls algorithm is determined by two steps: (*i*) picking the pivots (line 3), and (*ii*) building the clusters (line 4). Choosing the pivots requires $\mathcal{O}(m \log n)$ time, where $n = |V|$ and $m = |E|$, as selecting random edges can be implemented by building a priority queue of edges with random priorities, and subsequently removing edges; each edge is removed once from the priority queue, whether it is selected as pivot or not. Building a single cluster C, instead, requires to access all neighbors of the pivot edge (u, v). As the current cluster is removed from the set of uncovered objects at the end of each iteration, the neighbors of any pivot are not considered again in the remainder of the algorithm. Thus, the step of selecting the objects to be included into the current clusters requires visiting each edge at most once; therefore, the process takes $\mathcal{O}(m)$ time. In conclusion, we can state that the computational complexity of the Chromatic Balls algorithm is $\mathcal{O}(m \log n)$.

Algorithm 1 Chromatic Balls

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

1: $V' \leftarrow V$; $i \leftarrow 1$
2: **while** there exist $u, v \in V'$ such that $(u, v) \in E$ **do**
3: randomly pick $u, v \in V'$ such that $(u, v) \in E$
4: $C \leftarrow \{u, v\} \cup \{x \in V' \mid \ell(u, x) = \ell(v, x) = \ell(u, v)\}$
5: **for all** $x \in C$ **do** $\mathcal{C}(x) \leftarrow i$
6: $c\ell(i) = \ell(u, v)$
7: $V' \leftarrow V' \setminus C$; $i \leftarrow i + 1$
8: **for all** $x \in V'$ **do**
9: $\mathcal{C}(x) \leftarrow i$
10: $c\ell(i) \leftarrow$ a random label from L
11: $i \leftarrow i + 1$

3.1 Theoretical analysis

We analyze next the quality of the solutions obtained by Chromatic Balls. Our main result, given in Theorem 1, shows that the approximation guarantee of the algorithm depends on the number of *bad triplets* incident to a pair of objects in the input dataset. The notion of bad triplet is defined below; however, here we note that this result gives a constant-factor guarantee for bounded-degree graphs.

Even though the Chromatic Balls algorithm is similar to the Balls algorithm, which can be shown to provide a constant-factor approximation guarantee for general graphs too, the theoretical analysis of Chromatic Balls is much more complicated and requires several additional and nontrivial arguments. Due to the limited space of this paper, we report next only an outline of our analysis. Further details, including complete proofs, can be found in an extended technical report.[1]

We begin our analysis by defining special types of triplets and quadruples among the vertices of the graph.

Definition 1 (SC-triplet) *We say that* $\{x, y, z\}$ *is a* same-color triplet *(SC-triplet) if the induced triangle is monochromatic, i.e.,* $\ell(x, y) = \ell(x, z) = \ell(y, z) \neq l_0$.

Definition 2 (B-triplet) *We say that* $\{x, y, z\}$ *is a* bad triplet *(B-triplet) if the induced triangle is non-monochromatic and it has at most one pair labeled with* l_0.

Definition 3 (B-quadruple) *A* Bad-quadruple *is a set* $\{x, y, z, w\} \subseteq V$ *that contains at least one SC-triplet and at least one B-triplet.*

Note that, according to the cost function of our problem as defined in Equation (2), there is no way to partition a B-triplet without paying any cost. Next we define the notions of *hitting* and *d-hitting*.

Definition 4 (hitting) *Consider a pair of objects* (x, y) *and a triplet* t*, which can be either SC-triplet or B-triplet. We say that* t hits (x, y) *if* $x \in t$ *and* $y \in t$*. Additionally, if*

[1] `http://francescobonchi.com/CCC.pdf`

q is a B-quadruple, we say that q hits *(x, y) if $x \in q$, $y \in q$, and there exists $z \in q$ such that $\{x, y, z\}$ is a B-triplet.*

Definition 5 (d-hitting) *Given any pair of objects (x, y) and any B-quadruple $q = \{x, y, z, w\}$, we say that q* deeply hits *(*d-hits*) (x, y) if q hits (x, y) and either $\{x, z, w\}$ or $\{y, z, w\}$ is an SC-triplet.*

In reference to the above notions, we hereinafter denote by $\mathcal{S}$, $\mathcal{T}$, and $\mathcal{Q}$ the sets of all SC-triplets, B-triplets, and B-quadruples for an instance of our problem. Moreover, given a pair $(x, y) \in V \times V$ we define the following sets: $\mathcal{T}_{xy} \subseteq \mathcal{T}$ denotes the set of all B-triplets in $\mathcal{T}$ that hit (x, y); $\mathcal{Q}_{xy} \subseteq \mathcal{Q}$ denotes the set of all B-quadruples in $\mathcal{Q}$ that hit (x, y); $\mathcal{Q}^d_{xy} \subseteq \mathcal{Q}_{xy} \subseteq \mathcal{Q}$ denotes the set of all B-quadruples in $\mathcal{Q}$ that d-hit (x, y).

Let us now consider some events that may arise during the execution of the Chromatic Balls algorithm. Given an object $x \in V$, $P^{(i)}_x$ denotes the event "*x is chosen as pivot in the i-th iteration.*" Given a set $\{x_1, \ldots, x_n\} \subseteq V$, with $n \geq 2$, $A^{(i)}_{x_1 \cdots x_n}$ denotes the event "*all objects $x_1, \ldots, x_n$ enter the i-th iteration of the algorithm, while two of them are chosen as pivot in the same iteration.*"

Additionally, the events $T^{(i)}_{z|xy}$ and $Q^{(i)}_{zw|xy}$ are defined in reference to a pair (x, y). Given a B-triplet $\{x, y, z\} \in \mathcal{T}_{xy}$, $T^{(i)}_{z|xy}$ denotes the event "*$A^{(i)}_{xyz}$ occurs while x and y are not chosen both as pivots in the i-th iteration.*" Given a B-quadruple $\{x, y, z, w\} \in \mathcal{Q}^d_{xy}$, $Q^{(i)}_{zw|xy}$ denotes the event "*$A^{(i)}_{xyzw}$ occurs while neither x nor y are chosen as pivots in i-th iteration.*"

For the events $A^{(i)}_{x_1 \cdots x_n}$, $T^{(i)}_{z|xy}$, and $Q^{(i)}_{zw|xy}$, defined above, we also consider their counterparts that assert that the events occur at *some* iteration i. For instance, $A_{x_1 \cdots x_n}$ denotes the event "*$A^{(i)}_{x_1 \cdots x_n}$ happens at some iteration i*," while $T_{z|xy}$ and $Q_{zw|xy}$ are defined analogously. Formally:

$$A_{x_1 \cdots x_n} \Leftrightarrow \bigvee_i A^{(i)}_{x_1 \cdots x_n}, \tag{3}$$

$$T_{z|xy} \Leftrightarrow \bigvee_i T^{(i)}_{z|xy} \Leftrightarrow \bigvee_i \left(A^{(i)}_{xyz} \wedge \neg \left(P^{(i)}_x \wedge P^{(i)}_y \right) \right), \tag{4}$$

$$Q_{zw|xy} \Leftrightarrow \bigvee_i Q^{(i)}_{zw|xy} \Leftrightarrow \bigvee_i \left(A^{(i)}_{xyzw} \wedge \neg P^{(i)}_x \wedge \neg P^{(i)}_y \right). \tag{5}$$

As reported in the next two lemmas, the probabilities of the events $T_{z|xy}$ and $Q_{zw|xy}$ can be expressed in terms of the probabilities of the events A_{xyz} and A_{xyzw}.

Lemma 1 *Given a pair $(x, y) \in V \times V$ and a B-triplet $\{x, y, z\} \in \mathcal{T}_{xy}$, it holds that $\frac{1}{2} \Pr[A_{xyz}] \leq \Pr[T_{z|xy}] \leq \Pr[A_{xyz}]$.*

Lemma 2 *Given a pair $(x, y) \in V \times V$ and a B-quadruple $\{x, y, z, w\} \in \mathcal{Q}^d_{xy}$, it holds that $\frac{1}{6} \Pr[A_{xyzw}] \leq \Pr[Q_{zw|xy}] \leq \frac{1}{4} \Pr[A_{xyzw}]$.*

Analyzing carefully the probabilities of events $T_{z|xy}$ and $Q_{zw|xy}$ is crucial for deriving the desired approximation factor, as shown next.

We consider an instance $G = (V, E, L, \ell)$ of our problem and rewrite the cost function in Equation (2) as sum of the costs paid by any single pair (x, y). 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:

$$c(G) = \sum_{(x,y) \in V \times V} c_{xy}(G), \tag{6}$$

where $c_{xy}(G)$ denotes the aforementioned contribution of the pair (x, y) to the total cost. Moreover, let $\mathrm{E}[c(G)]$ denote the expected cost of Chromatic Balls over the random choices made by the algorithm. By the linearity of expectation, the expected cost $\mathrm{E}[c(G)]$ can be expressed as

$$\mathrm{E}[c(G)] = \sum_{(x,y) \in V \times V} \mathrm{E}\left[c_{xy}(G)\right]. \tag{7}$$

Finally, let $c^*(G)$ be the cost of the optimal solution on G.

To derive an approximation factor $r(G)$ on the performance of the Chromatic Balls algorithm, we look for an upper bound $Ub(G)$ on the expected cost $\mathrm{E}[c(G)]$ of the algorithm, and a lower bound $Lb(G)$ on the cost $c^*(G)$ of the optimal solution, so that

$$\frac{\mathrm{E}[c(G)]}{c^*(G)} \leq \frac{Ub(G)}{Lb(G)} = r(G). \tag{8}$$

We next show how to derive such upper and lower bounds.

Deriving the upper bound $Ub(G)$. For a pair (x, y) we define the collection of events $\Omega_{xy} = \{T_{z|xy} \mid \{x, y, z\} \in \mathcal{T}_{xy}\} \cup \{Q_{zw|xy} \mid \{x, y, z, w\} \in \mathcal{Q}^d_{xy}\}$. As the following two lemmas show, if pair (x, y) contributes to the cost paid by the algorithm, then exactly one of the events in Ω_{xy} occurs.

Lemma 3 *If $c_{xy}(\mathcal{C}, c\ell, G) > 0$ then at least one of the events in Ω_{xy} occurs.*

Lemma 4 *The events within the collection Ω_{xy} are disjoint.*

Combining Lemmas 3 and 4 with the expressions of the probabilities of the events $T_{z|xy}$ (Lemma 1) and $Q_{zw|xy}$ (Lemma 2) we can derive an upper bound on the expected contribution $\mathrm{E}[c_{xy}(G)]$ of a pair (x, y) to the total cost.

Lemma 5 *For a pair $(x, y) \in V \times V$ the following bound holds.*

$$\mathrm{E}[c_{xy}(G)] \leq \sum_{\{x,y,z\} \in \mathcal{T}_{xy}} \Pr[A_{xyz}] + \sum_{\{x,y,z,w\} \in \mathcal{Q}^d_{xy}} \frac{1}{4} \Pr[A_{xyzw}].$$

The bound in Lemma 5 together with Equation (7) can be used to give the desired (upper) bound on the overall expected cost $\mathrm{E}[c(G)]$.

Lemma 6 *The expected cost $\mathrm{E}[c(G)]$ of the Chromatic Balls algorithm can be bounded as follows*

$$\mathrm{E}[c(G)] \leq Ub(G) = \sum_{\{x,y,z\} \in \mathcal{T}} \left(3 \Pr[A_{xyz}] + \frac{3}{4} X_{xyz} + \frac{1}{2} Y_{xyz} \right),$$

where:

$$X_{xyz} = \sum_{w \in W_{xyz}} \frac{\Pr[A_{xyzw}]}{\tau_{xyzw}}, \; Y_{xyz} = Y^{xy}_{xyz} + Y^{xz}_{xyz} + Y^{yz}_{xyz},$$

$$Y^{xy}_{xyz} = \sum_{w \in W^{xy}_{xyz}} \frac{\Pr[A_{xyzw}]}{\tau_{xyzw}}, \; Y^{xz}_{xyz} = \sum_{w \in W^{xz}_{xyz}} \frac{\Pr[A_{xyzw}]}{\tau_{xyzw}},$$

$$\text{and} \;\; Y^{yz}_{xyz} = \sum_{w \in W^{yz}_{xyz}} \frac{\Pr[A_{xyzw}]}{\tau_{xyzw}}.$$

Finally, τ_{xyzw} denotes the number of B-triplets contained in any B-quadruple $\{x, y, z, w\}$.

Deriving the lower bound $Lb(G)$. Recalling that a B-triplet incurs a non-zero cost in any solution, a lower bound on the cost of the optimal solution $c^*(G)$ can be obtained by counting the number of disjoint B-triplets in the input. Considering the set $\mathcal{T}$ of B-triplets we can restate the following result of Ailon et al. [1] that provides a lower bound on the optimal by "fractionally assigning" all pairs of objects in $V \times V$ to the triplets in $\mathcal{T}$.

Lemma 7 (Ailon et al. [1]) *Let $\{\alpha_{xyz} \mid \{x,y,z\} \in \mathcal{T}\}$ be any assignment of nonnegative weights to the B-triplets in $\mathcal{T}$ satisfying $\sum_{\{x,y,z\}\in\mathcal{T}_{x'y'}} \alpha_{xyz} \leq 1$ for all $(x',y') \in V \times V$. It holds that $c^*(G) \geq \sum_{\{x,y,z\}\in\mathcal{T}} \alpha_{xyz}$.*

We can then obtain a lower bound on the optimal solution by finding a suitable set of weights α_{xyz} that satisfies the conditions of the previous lemma. We derive such a set of weights in the following further lemma.

Lemma 8 *For any pair $(x,y) \in V \times V$ the following condition holds.*

$$\sum_{\{x,y,z\}\in\mathcal{T}_{xy}} \frac{1}{1+|\mathcal{T}_{xy}|}\left(\frac{1}{2}\Pr\left[A_{xyz}\right] + \frac{1}{6}X_{xyz} + \frac{1}{6}Y_{xyz}\right) \leq 1.$$

Thus, combining Lemmas 7 and 8, we can obtain the desired lower bound $Lb(G)$ as follows.

Lemma 9 *The cost $c^*(G)$ of the optimal solution on any input instance G is lower bounded as follows*

$$c^*(G) \;\geq\; Lb(G) \;=$$
$$= \sum_{\{x,y,z\}\in\mathcal{T}} \frac{1}{1+t_{max}}\left(\frac{1}{2}\Pr\left[A_{xyz}\right] + \frac{1}{6}X_{xyz} + \frac{1}{6}Y_{xyz}\right),$$

where $t_{max} = \max_{(x,y)\in V\times V} |\mathcal{T}_{xy}|$ is the maximum number of B-triplets that hit a pair of objects.

The approximation ratio $r(G)$. The upper and lower bounds obtained in Lemmas 6 and 9 are at the basis if the final form of the approximation ratio of Chromatic Balls.

Theorem 1 *The approximation ratio of the Chromatic Balls algorithm on any input instance G is*

$$r(G) = \frac{\mathrm{E}[c(G)]}{c^*(G)} \leq 6\ (1+t_{max}),$$

where $t_{max} = \max_{(x,y)\in V\times V} |\mathcal{T}_{xy}|$ is the maximum number of B-triplets that hit a pair of objects.

Theorem 1 shows that the approximation factor of the Chromatic Balls algorithm is bounded by the maximum number t_{max} of B-triplets that hit a pair of objects. The result is meaningful as it 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 1 *The approximation ratio of the Chromatic Balls algorithm on any input instance G is*

$$r(G) \leq 6\,(2D_{max} - 1),$$

where $D_{max} = \max_{x\in V} |\{y \mid y \in V \wedge \ell(x,y) \neq l_0\}|$ is the maximum degree in the problem instance.

4. OTHER ALGORITHMS

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 attempts to overcome some weaknesses of Chromatic Balls by employing two heuristics, one for pivot selection and one for cluster selection. The second one is an alternating minimization method that is designed to optimize directly the objective function.

4.1 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 *Consider the graph in Figure 2: it has a fairly evident* `green` *cluster formed by vertices {U,V,R,X,Y,W,Z}.*

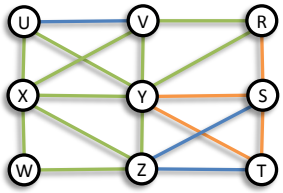


Figure 2: An example of an edge-labeled graph.

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 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 $d(x,l)$ be the number of edges incident to x having label l. Also, we denote by $\Delta(x) = \max_{l\in L} d(x,l)$, and $\lambda(x) = \arg\max_{l\in L} d(x,l)$. Lazy Chromatic Balls differs from Chromatic Balls in two ways:

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

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

Example 4 *Consider again the example in 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 first pivot vertex u by Lazy Chromatic Balls with higher probability than the remaining vertices. Suppose that*

Algorithm 2 Alternating Minimization (AM)

Input: Edge-labeled graph $G = (V, E, L, \ell)$; 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 $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_N]$ and $\mathbf{C} = [\mathbf{c}_1, \ldots, \mathbf{c}_K]$ at random
2: **repeat**
3: for all $x \in V$ compute optimal $\mathbf{a}_x$ according to Proposition 1
4: for all $k \in [1..K]$ compute optimal $\mathbf{c}_k$ according to Proposition 2
5: **until** neither $\mathbf{A}$ nor $\mathbf{C}$ changed

X is picked up, i.e., u = X. Given this choice, the second pivot v is chosen among the neighbors of X with probability proportional to $d(v, \lambda(u))$, 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 v, thus making (X,Y) the selected pivot edge. Afterwards,* Lazy Chromatic Balls *adds to the being 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.*

Computational complexity. Like Chromatic Balls, the running time of the Lazy Chromatic Balls algorithm is determined by picking the pivots and building the various clusters. Picking the first pivot u can be implemented with a priority queue with priorities $\Delta \times rnd$, where rnd is a random number. This requires computing Δ for all objects, which takes $\mathcal{O}(nh + m)$ (where $h = |L|$). Managing the priority queue itself requires instead $\mathcal{O}(n \log n)$, as each object is put into/removed from the queue only once during the execution of the algorithm. Given u, the second pivot v is selected by probing all (non-chosen) neighbors of u. This takes $\mathcal{O}(m)$ time, as for each pivot u, its neighbors are accessed only once throughout the execution of the algorithm. Finally, building the current cluster takes $\mathcal{O}(m)$ time, as it requires a visit of the graph, where each edge is accessed $\mathcal{O}(1)$ times. In conclusion, the computational complexity of Lazy Chromatic Balls is $\mathcal{O}(n(\log n + h) + m)$, which, for small h, is better than the complexity of Chromatic Balls.

4.2 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 could be desired having an output clustering composed by a pre-specified number K of clusters. To this purpose, we present here an algorithm based on the *alternating minimization* paradigm [7], that receives in input the number K of output clusters and attempts to minimize Equation (2) directly. The pseudocode of the proposed algorithm, called Alternating Minimization, is given in Algorithm 2.

In a nutshell, AM tries to produce a solution by alternating between two optimization steps. In the first step the algorithm finds the best cluster assignment for every $x \in V$ given the assignments of every other $y \in V$ and the current cluster labels. In the second step, it finds the best label for every cluster given the current assignment of objects to clusters. Below we show that both steps can be solved optimally. As a consequence the value of Equation (2) is guaranteed to decrease in every step, until convergence. Finding the global minimum is obviously hard, but the algorithm is guaranteed to converge to a local optimum.

Definitions. For presentation sake, we adopt matrix notation. We denote matrices by uppercase boldface romans and vectors by lowercase boldface romans. We write $\mathbf{X}_{ij}$ for the (i, j) coordinate of matrix $\mathbf{X}$, and $\mathbf{x}(i)$ for the i-th coordinate of vector $\mathbf{x}$.

The parameter space of Problem 2 consists of a *cluster assignment* for every object $x \in V$, given by the binary matrix $\mathbf{A}$, and a *label assignment* for every cluster $k \in \{1, \ldots, K\}$, given by the binary matrix $\mathbf{C}$. We have $\mathbf{A}_{kx} = 1$ when object x is assigned to cluster k, and $\mathbf{A}_{kx} = 0$ otherwise. Similarly, we set $\mathbf{C}_{lk} = 1$ when label l is assigned to cluster k, and $\mathbf{C}_{lk} = 0$ otherwise. Since every object must belong to one and only one cluster, and every cluster must have one and only one label assigned, both $\mathbf{A}$ and $\mathbf{C}$ are constrained to consist of all zeros with a single 1 on every column. Denote by $\mathbf{a}_x$ the column of $\mathbf{A}$ that corresponds to object x.

The input is represented by a set of binary matrices, with a matrix $\mathbf{Z}_x$ for every $x \in V$. These matrices encode the labeling function ℓ as follows. Let $\mathbf{z}_{xy}$ denote the column of $\mathbf{Z}_x$ that corresponds to the object $y \in V$. We have $\mathbf{z}_{xy}(l) = 1$ if and only if $\ell(x, y) = l$, otherwise $\mathbf{z}_{xy}(l) = 0$. Every $\mathbf{Z}_x$ consists thus of zeros, with exactly one 1 on every column. Finally, denote by $\mathbf{b}$ a special binary vector where $\mathbf{b}(l) = 1$ when $l = l_0$ and $\mathbf{b}(l) = 0$ otherwise. We have then $\mathbf{z}_{xy}^{\mathrm{T}}\mathbf{b} = 1$ if and only if $\ell(x, y) = l_0$.

The above formulation of the problem assumes that the input is represented by many large matrices. Note however that this representation is only conceptual. In the actual implementation we do not have to materialize these matrices and we can represent the input with the minimal amount of space required, as shown next. The benefit of our formulation is that it allows to write our objective function and our optimization process using linear-algebra operations, and argue about the optimality of the local optimization steps.

Optimal cluster assignment. Denote by N_{xk}^{-} the number of objects $y \in V$ in cluster k that have $\ell(x, y) = l_0$. Since $\ell(x, y) = l_0 \Leftrightarrow \mathbf{z}_{xy}\mathbf{b} = 1$, we have $N_{xk}^{-} = (\mathbf{A}\mathbf{Z}_x\mathbf{b})(k)$. Similarly, let N_{xk}^{+} denote the number of objects $y \in V$ in cluster k that have $\ell(x, y) = c\ell(k)$. Since $y \in k$, we have $\ell(x, y) = c\ell(k) \Leftrightarrow \mathbf{z}_{xy}\mathbf{C}\mathbf{a}_y = 1$ and can write $N_{xk}^{+} = (\mathbf{A}\mathbf{w}_x)(k)$, where $\mathbf{w}_x = [\mathbf{z}_{x1}^{\mathrm{T}}\mathbf{C}\mathbf{a}_1 \ldots \mathbf{z}_{xn}^{\mathrm{T}}\mathbf{C}\mathbf{a}_n]$.

Proposition 1 *The optimal cluster assignment for $x \in V$ given $\mathbf{A}$ and $\mathbf{C}$ is $k^* = \arg\min_k N_{xk}^{-} - N_{xk}^{+}$.*

Proof. We can rewrite Equation (2) as follows:

$$\sum_{x,y} \mathbf{a}_x^{\mathrm{T}}\mathbf{a}_y(1 - \mathbf{z}_{xy}^{\mathrm{T}}\mathbf{C}\mathbf{a}_y) + (1 - \mathbf{a}_x^{\mathrm{T}}\mathbf{a}_y)(1 - \mathbf{z}_{xy}^{\mathrm{T}}\mathbf{b}) = \quad (9)$$
$$= \sum_x \mathbf{a}_x^{\mathrm{T}}\mathbf{A}(\mathbb{1} - \mathbf{w}_x) + (\mathbb{1}^{\mathrm{T}} - \mathbf{a}_x^{\mathrm{T}}\mathbf{A})(\mathbb{1} - \mathbf{Z}_x^{\mathrm{T}}\mathbf{b}),$$

where $\mathbf{w}_x$ is defined as above, and $\mathbb{1}$ denotes the $|V|$-dimensional vector of all 1s. Terms that correspond to a fixed $x \in V$ further simplify to

$$\mathbf{a}_x^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{Z}_x\mathbf{b} - \mathbf{a}_x^{\mathrm{T}}\mathbf{A}\mathbf{w}_x + d_x,$$

where the constant $d_x = \mathbb{1}^{\mathrm{T}}\mathbb{1} - \mathbb{1}^{\mathrm{T}}\mathbf{Z}_x^{\mathrm{T}}\mathbf{b}$ is the "degree" of object x, the number of objects $y \in V$ where $\ell(x, y) \neq l_0$.

Since we must assign exactly one cluster for x, the above expression is minimized simply by assigning x to the cluster k that minimizes $(\mathbf{A}\mathbf{Z}_x^{\mathrm{T}}\mathbf{b})(k) - (\mathbf{A}\mathbf{w}_x)(k) = N_{xk}^- - N_{xk}^+$.

The result is quite intuitive. The best cluster for x is the one having the least "push" in terms of l_0 connections, and the most "pull" given by connections having the appropriate label. However, evaluating N_{xk}^- in practice is very slow, as it involves checking all l_0 connections of x. Ideally the update rule should only require access to edges having some label other than l_0. This is easy to achieve, however.

Let N_{xk}^0 denote the remaining objects in cluster k, that is, those with $\ell(x,y) \neq c\ell(k) \neq l_0$. Also, let S_k denote the size of cluster k. Clearly we have $S_k = N_{xk}^+ + N_{xk}^0 + N_{xk}^-$ for every $x \in V$. Using this we obtain $N_{xk}^- - N_{xk}^+ = S_k - 2N_{xk}^+ - N_{xk}^0$, which is much faster to evaluate.

Optimal label assignment. The update rule for the cluster label is intuitive as well. Denote by E_k the number of ordered (x,y) pairs so that both x and y belong to cluster k, and $\ell(x,y) = c\ell(k)$.

Proposition 2 *The optimal label assignment for cluster k given $\mathbf{A}$ is $l^* = \arg\min_l S_k^2 - E_k$.*

Proof. We can partition the cost in Equation (9) as a sum over clusters. That is, for a fixed cluster k we sum only over those x and y that belong both to k. Also, the second term in Equation (9) does not depend on $\mathbf{C}$ and can therefore be omitted. This leaves us with the sum

$$\sum_{x\in k, y\in k} (1 - \mathbf{z}_{xy}^{\mathrm{T}}\mathbf{C}\mathbf{a}_y),$$

where we can replace $\mathbf{C}\mathbf{a}_y$ with the binary vector $\mathbf{c}_k$ that indicates the label assigned to cluster k. Clearly we have $\sum_{x\in k, y\in k} 1 = S_k^2$, and it is easy to see that $\sum_{x\in k, y\in k} \mathbf{z}_{xy}\mathbf{c}_k$ counts all (x,y) pairs having the same label that is currently assigned to k, which is by definition equal to E_k.

This means that the optimal label for cluster k is simply the label shared by the majority of the pairs in k.

Computational complexity. The running time of Alternating Minimization depends on the (optimal) cluster and label assignment steps. Cluster assignment requires two sub-steps: evaluating $S_k - 2N_{xk}^+ - N_{xk}^0$ for each vertex and cluster, which can be performed in $\mathcal{O}(m)$ by a simple visit of the input graph, and looking at all clusters to choose the best one for each vertex, which clearly takes $\mathcal{O}(Kn)$. Label assignment requires to compute the number of intra-cluster edge labels for each cluster k and label l. This takes $\mathcal{O}(m)$, as it can be performed, again, by visiting the input graph. Then, the assignment of labels to clusters by evaluating $S_k^2 - E_k$ can be performed in $\mathcal{O}(Kh)$. In conclusion, as usually $h = |L| \ll n$, the computational complexity of Alternating Minimization is $\mathcal{O}(s(Kn + m))$, where s is the number of iterations to convergence.

5. EXPERIMENTAL EVALUATION

In this section, we report our empirical assessments. We experiment with all three proposed algorithms, Chromatic Balls, Lazy Chromatic Balls, and Alternating Minimization, to which we refer by CB, LCB, and AM, respectively. We also evaluate the performance of the baseline described in the Introduction, namely the "standard" Balls algorithm [1] that ignores colors. We refer to this baseline as B. All measurements reported are averaged over 50 runs.

Algorithm 3 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 $G = (V, E, L, \ell)$

$V \leftarrow [1, n]$, $E \leftarrow \emptyset$, $L \leftarrow \{l_1, \ldots, l_h\}$
assign each vertex $x \in V$ to a randomly selected cluster
assign to each cluster a randomly selected label from L
for all pairs $(x, y) \in V \times V$ **do**
  pick 3 random numbers r_1, r_2, r_3 ranging within $[0, 1]$
  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
        $E \leftarrow E \cup (x, y)$
        $\ell(x, y) \leftarrow c\ell(\mathcal{C}(x))$;
  else if $r_3 < q$ **then**
    $E \leftarrow E \cup (x, y)$
    $\ell(x, y) \leftarrow$ a random label from L

5.1 Experiments on synthetic data

We evaluate our algorithms on synthetic datasets generated by the process outlined in Algorithm 3. 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.

The initial assignment of objects and labels to clusters can be interpreted 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 the $\hat{k}$-th cluster of $\widetilde{\mathcal{C}}$ and the k-th cluster 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 $F \in [0, 1]$.

We generate datasets with a fixed number of objects ($n = 1000$), and we vary (i) the noise level (controlled by playing with p, q, and w); (ii) the number of labels h; and (iii) the number of clusters K in the ground truth. Even though we perform tests by varying all parameters p, q, and w, due to space limitations we only report results obtained for varying q and keeping p and w equal to 0.5.

For the number of clusters required as input for the AM algorithm, we consider two options: the average number of clusters produced by the CB algorithm, and the number of clusters in the ground truth. We refer to these two settings by AM and AM*, respectively. In Figure 3 we report the performance of our algorithms in terms of F-measure, as well as solution cost (Equation (2)).

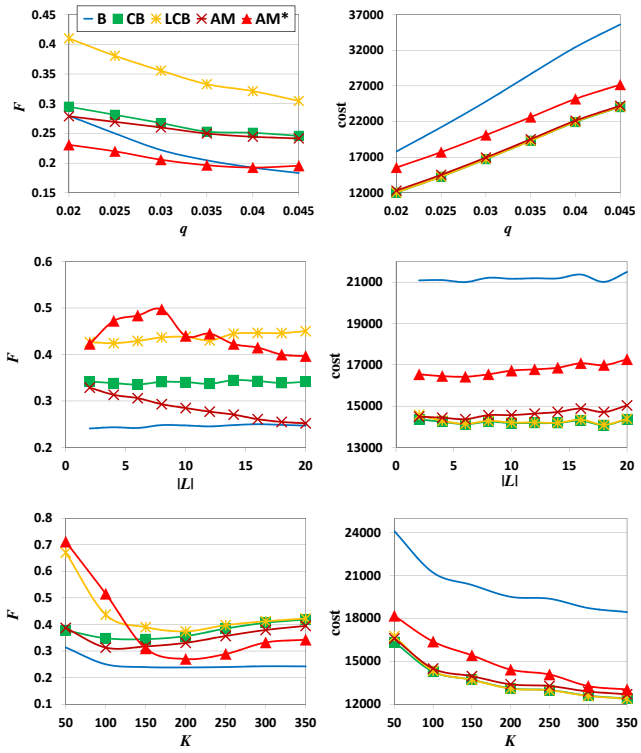


Figure 3: Accuracy on synthetic datasets in terms of F-measure (left) and solution cost (right), by varying level of noise (1st row), number of labels (2nd row), and number of ground-truth clusters (3rd row).

All trends observed by varying the parameters q, h, 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, by setting a larger K the difference tends to disappear.

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.

5.2 Experiments on real data

We experiment with three real datasets (Table 1).

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. The PPI datasets are usually very sparse, therefore, we keep only the 30-core of the entire network, i.e., we recursively remove the vertices with degree less than 30 until a fix-point has been reached.

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. For edges with multiple labels we picked one label at random from the available ones. The dataset has been compiled by Tang et al. [14] and it is available at `http://www.public.asu.edu/~ltang9/`.

Table 1: Characteristics of real data. n: number of vertices; m: number of edges; d: average degree; $|L|$: number of labels; c: clustering coefficient.

dataset	n	m	d	$\|L\|$	c
String	18 152	401 582	44.25	4	0.731
Youtube	15 088	19 923 067	2 640.92	5	0.495
DBLP	312 416	2 110 470	13.51	100	0.204

DBLP. We obtain a recent snapshot of the 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) [5] to automatically identify 100 topics on each edge. After LDA topic-modeling, for each edge, we assign its most prominent topic discovered as edge label.

Results. Table 2 summarizes the results obtained on real data. Like in synthetic data, all proposed algorithms clearly outperform the baseline B. CB is the best method on Youtube and DBLP, achieving up to 27.74% of 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 the runtime, we observe that the baseline is faster than the proposed methods, as expected. This is mainly due to a smaller complexity in choosing vertex pivots compared to choosing edge pivots. However, all proposed methods remain very efficient, as they take a few seconds (CB and LCB) or minutes (AM) on large and dense graphs like Youtube and DBLP. All runtimes comply with the computational complexity analysis reported previously. Indeed, AM is the slowest method, mostly due to the typically high number of iterations needed to convergence, while LCB is faster than CB, especially on dense datasets like Youtube.

Finally, Figure 4 shows an example 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".

6. RELATED WORK

Edge-labeled graphs and multidimensional networks. Graphs in which edges are labeled with a type of relation occurring among the connected vertices are receiving increasingly attention. To the best of our knowledge no previous work has investigated the problem of clustering in such graphs. The problems studied so far on this kind of graphs are mainly on label-constrained reachability queries [8, 10, 12, 16], whose main goal is to answer whether a vertex u can reach vertex v trough a path whose edge labels belong to a given set. Clustering has been studied, instead, in so called *multidimensional networks*, i.e., networks defined as a col-

Table 2: Results on real datasets: average cost, runtime (s), and average number of output clusters

	cost				runtime (s)				#clusters			
dataset	B	CB	LCB	AM	B	CB	LCB	AM	B	CB	LCB	AM
String	163 305	160 060	155 881	156 976	0.5	1.4	1.3	21.0	1 086	1 451	784	1 451
Youtube	23 550 213	18 956 000	22 644 858	19 670 899	22.4	117.8	40.5	1 038.9	568	1 078	672	1 078
DBLP	2 260 065	1 633 149	1 678 714	2 018 952	4.3	10.2	5.5	2 116.1	66 276	123 197	99 948	123 197

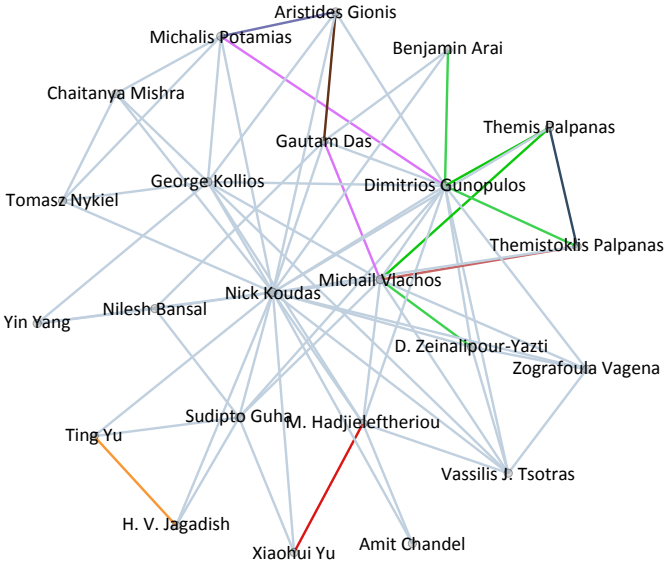


Figure 4: An example cluster from DBLP.

lection of multiple networks over the same set of actors. In our jargon these are simply graphs where each edge can have more than one color [4, 13, 15]. Although the input of that problem might seem close to ours, the objective is semantically far away. In clustering multidimensional networks, the objective is to find a partitioning of vertices which is meaningful and relevant in *all dimensions at the same time*. Taking again the colors metaphor, in that setting is a clustering is considered as good if it makes sense in the `green` network *and* as well as the `red` network, and so on. In our work, we are rather interested in finding groups of objects that induce color-coherent clusters while looking at all the colors together.

Correlation Clustering. The problem of CORRELATION-CLUSTERING was first defined by Bansal et al. [3] in its binary version. Ailon et al. [1] proposed the Balls algorithm that achieves expected approximation factor 5 if the weights obey the probability condition. If the weights X_{ij} obey also the triangle inequality, then the algorithm achieves expected approximation factor 2. Giotis and Guruswami [9] consider correlation clustering when the number of clusters is given, while Ailon and Liberty [2] 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. We recently extended correlation clustering to allow overlaps, i.e., objects belonging to more than one cluster [6].

7. CONCLUSIONS

In this paper, we introduce a variant of the correlation-clustering problem, in which the pairwise relations between objects are categorical. The problem has interesting applications, such as clustering social networks where individuals are connected with different types of relations, or clustering protein networks, where proteins are associated with different types of interactions. We propose three algorithms that we evaluate on synthetic and real datasets.

Our problem is a novel clustering formulation well-suited for mining multi-labeled and heterogeneous datasets that are becoming increasingly common. We believe that there are many interesting extensions and fruitful future research directions. For example, we would like to extend the problem formulation in order to capture overlapping clusters as well as multiple-labeled edges.

Acknowledgements. This research was partially supported by the Torres Quevedo Program of the Spanish Ministry of Science and Innovation, co-funded by the European Social Fund, and by the Spanish Centre for the Development of Industrial Technology under the CENIT program, project CEN-20101037, "Social Media" (`http://www.cenitsocialmedia.es/`).

8. REFERENCES

[1] N. Ailon, M. Charikar, and A. Newman. Aggregating inconsistent information: Ranking and clustering. *JACM*, 55:23:1–23:27, 2008.

[2] N. Ailon and E. Liberty. Correlation clustering revisited: The "true" cost of error minimization problems. In *ICALP*, 2009.

[3] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine Learning*, 56(1–3), 2004.

[4] M. Berlingerio, M. Coscia, and F. Giannotti. Finding and Characterizing Communities in Multidimensional Networks. In *ASONAM*, pages 490–494, 2011.

[5] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *JMLR*, 3:993–1022, 2003.

[6] F. Bonchi, A. Gionis, and A. Ukkonen. Overlapping correlation clustering. In *ICDM*, pages 51–60, 2011.

[7] I. Csiszar and G. Tusnady. Information geometry and alternating minimization procedures. *Statistics and Decisions*, 1984.

[8] W. Fan, J. Li, S. Ma, N. Tang, and Y. Wu. Adding Regular Expressions to Graph Reachability and Pattern Queries. In *ICDE*, pages 39–50, 2011.

[9] I. Giotis and V. Guruswami. Correlation clustering with a fixed number of clusters. In *SODA*, 2006.

[10] R. J. H. Hong, H. Wang, N. Ruan, and Y. Xiang. Computing Label-Constraint Reachability in Graph Databases. In *SIGMOD*, pages 123–134, 2010.

[11] C. Lin et al. Clustering methods in protein-protein interaction networks. In *Knowledge Discovery in Bioinformatics: Techniques, Methods and Application*.

[12] M. Rice and V. J. Tsotras. Graph indexing of road networks for shortest path queries with label restrictions. *PVLDB*, 4:69–80, 2010.

[13] M. Rocklin and A. Pinar. On Clustering on Graphs with Multiple Edge Types. In *WAW*, pages 38–49, 2011.

[14] L. Tang, X. Wang, and H. Liu. Uncovering groups via heterogeneous interaction analysis. In *ICDM '09: Proceedings of IEEE International Conference on Data Mining*, pages 503–512, 2009.

[15] L. Tang, X. Wang, and H. Liu. Community detection via heterogeneous interaction analysis. *Data Mining and Knowledge Discovery*, pages 1–33, 2011.

[16] K. Xu, L. Zou, J. X. Yu, L. Chen, Y. Xiao, and D. Zhao. Answering Label-Constraint Reachability in Large Graphs. In *CIKM*, pages 1595–1600, 2011.