A Color-blind 3-Approximation for Chromatic Correlation Clustering and Improved Heuristics

Nicolas Klodt
Lars Seifert
Arthur Zahn
{firstname.lastname}@uni-potsdam.de
University Potsdam
Potsdam, Germany

Katrin Casel
Davis Issac
{firstname.lastname}@hpi.de
Hasso Plattner Institute
Potsdam, Germany

Tobias Friedrich
tobias.friedrich@uni-potsdam.de
University Potsdam
Hasso Plattner Institute
Potsdam, Germany

ABSTRACT

Chromatic Correlation Clustering (CCC) models clustering of objects with *categorical* pairwise relationships. The model can be viewed as clustering the vertices of a graph with edge-labels (colors). Bonchi et al. [KDD 2012] introduced it as a natural generalization of the well studied problem Correlation Clustering (CC), motivated by real-world applications from data-mining, social networks and bioinformatics. We give theoretical as well as practical contributions to the study of CCC.

Our main theoretical contribution is an alternative analysis of the famous Pivot algorithm for CC. We show that, when simply run color-blind, Pivot is also a linear time 3-approximation for CCC. The previous best theoretical results for CCC were a 4-approximation with a high-degree polynomial runtime and a linear time 11-approximation, both by Anava et al. [WWW 2015].

While this theoretical result justifies Pivot as a baseline comparison for other heuristics, its blunt color-blindness performs poorly in practice. We develop a color-sensitive, practical heuristic we call Greedy Expansion that empirically outperforms all heuristics proposed for CCC so far, both on real-world and synthetic instances.

Further, we propose a novel generalization of CCC allowing for multi-labelled edges. We argue that it is more suitable for many of the real-world applications and extend our results to this model.

CCS CONCEPTS

• **Information systems** → **Clustering**; • **Theory of computation** → **Approximation algorithms analysis**.

KEYWORDS

correlation clustering, edge-labeled graphs, approximation

ACM Reference Format:
Nicolas Klodt, Lars Seifert, Arthur Zahn, Katrin Casel, Davis Issac, and Tobias Friedrich. 2021. A Color-blind 3-Approximation for Chromatic Correlation Clustering and Improved Heuristics. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3447548.3467446

KDD '21, August 14–18, 2021, Virtual Event, Singapore

ACM ISBN 978-1-4503-8332-5/21/08.
https://doi.org/10.1145/3447548.3467446

1 INTRODUCTION

Clustering is one of the most important computational tasks in data science. Grouping similar objects has applications in most fields of science, technology and engineering. One of the widely used clustering methods is Correlation Clustering (CC) [5], which is suitable when the objects to be clustered are given with a pairwise similarity relationship. Indeed, in many applications we find such a scenario, usually represented as a graph with objects as vertices and edges as similarity relationship. This makes CC an indispensable tool in any data miner's toolkit [11]. The objective of CC is to minimize disagreements i.e. the inter-cluster differences and intra-cluster similarities. Since CC can be phrased as concise graph-theoretical problem and allows the use of combinatorial techniques, it has also attracted wide interest in theoretical computer science [10].

Correlation Clustering in its standard definition considers pairwise relationships in binary form. However, in many real-world applications, we are presented with also *categorical* information about relationships. As an example, consider a social network where similarity means *friendship*; there are different types of friendships such as schoolmates, colleagues, and family. It is desirable to cluster people into groups that have the same category of friendship between them. Motivated by this, Bonchi et al. [7] gave a generalized model for CC called Chromatic Correlation Clustering (CCC), that makes use of the categorical information. In this model, the input is an edge-colored graph where colors model categories of similarity, and the objective includes minimizing inter-cluster color differences in addition to the CC objective. The chromatic model has the additional benefit that each output cluster is associated with a category, e.g. in the social network example, the output will consist of schoolmate-, colleague-, and family-clusters.

Apart from social networks, CCC finds applications in many other fields. Biological networks also tend to have different types of interactions. For example, in protein-protein interaction networks, the association between proteins can be physical, co-localization, genetic and many other types [12]. Taking these types into account allows to interpret the resulting clusters as functional modules, protein complexes, etc. Another application is the clustering of authors

based on collaborations and research topics to help researchers navigate literature more easily [8].

1.1 Problem Statement

An instance $G = (V, E, col)$ of Chromatic Correlation Clustering (CCC) is an edge-colored graph G with vertex set V, edge set E, and color function $col\colon E \to \mathbb{N}$. We refer to $L = \text{image}(col)$ as the *set of colors* or *labels*. We use uv as short notation for $\{u, v\} \in \binom{V}{2}$.

A solution to a CCC instance $G = (V, E, col)$ is a clustering $Sol = (C, ccol)$ consisting of a partition of the vertices into clusters $C\colon V \to \mathbb{N}$ and a cluster color assignment $ccol\colon \text{range}(C) \to L$. The objective is to minimize the number of disagreements, where a vertex pair is considered to be a disagreement if it is a non-edge inside of a cluster (*intra-cluster disagreements*), an edge between clusters (*inter-cluster disagreements*), or an edge inside a cluster with a color different from the cluster color (*color disagreements*). Formally, the number of disagreements $d(G, Sol)$ is defined as

$$d_{uv}(G, Sol) = \begin{cases} 1, \text{ if } uv \notin E \wedge C(u) = C(v) \\ 1, \text{ if } uv \in E \wedge C(u) \neq C(v) \\ 1, \text{ if } uv \in E \wedge C(u) = C(v) \wedge col(uv) \neq ccol(C(u)) \\ 0, \text{ otherwise} \end{cases}$$

$$D(G, Sol) = \{uv \in \tbinom{V}{2} \mid d_{uv}(G, Sol) = 1\}$$
$$d(G, Sol) = |D(G, Sol)|$$

We refer to the objective value of an optimum solution with $d^*(G)$. In the above definitions, we omit the first argument G when the graph is clear from the context. Observe that the CC problem is a special case of CCC where $|L| = 1$.

For a vertex $v \in V$, we define the *primary color* of v as

$$p(v) := arg\max_{c \in L} |\{uv \in E \mid col(uv) = c\}|.$$

Note that we break ties arbitrarily here. We call an edge a *secondary edge* if it has a color different from the primary color of either of its incident vertices, and define the set

$$E_2 = \{uv \in E \mid col(uv) \neq p(u) \vee col(uv) \neq p(v)\}.$$

1.2 Our Results

We give a simple linear time 3-approximation algorithm for CCC. The previous best approximation guarantees were 4 in polynomial time and 11 in linear time, both by Anava et al. [4]. Our algorithm is surprisingly simple: it just ignores the colors of the edges and runs the well-known `Pivot` algorithm for CC. The key to our improved approximation ratio is a clever charging method for directly bounding the errors of `Pivot` by the errors of an optimum solution.

We improve the analysis of the monochromatic reduction algorithmic framework for CCC proposed by Anava et al. [4]. This framework reduces CCC to CC while incurring some error during the reduction, and then runs a CC algorithm. We better bound the error during the reduction and show that using an α approximate CC algorithm gives an $(\alpha + 2)$-approximate CCC algorithm, improving upon the $(3\alpha + 2)$-guarantee given by Anava et al. [4]. In particular, we prove that this framework with `Pivot`, called Reduce and Cluster (`RC`), is in fact a linear time 5-approximation improving upon the 11-approximate guarantee given by Anava et al. [4].

We propose an algorithm called Greedy Expansion (`GE`) and compare it experimentally to the previous heuristics for CCC, on synthetic and real-world datasets. We also compare `GE` with the monochromatic reduction framework of Anava et al. [4] when combined with `Vote` and `RMM`, which are two known heuristics for CC. To the best of our knowledge, monochromatic reduction combined with `Vote` and `RMM` has not been studied before. `GE` and the modified `Vote` and `RMM` significantly outperform all the previously known heuristics for CCC on all the synthetic and real-world datasets. Our `GE` heuristic also performs significantly better than `Vote` and `RMM` on most of the datasets and performs similarly on some of them.

We also propose a new multi-label model for CCC, where each edge is allowed to have a set of colors. Our model is based on the observation that many real-world data in fact has multiple labels on edges and we may loose valuable information while converting it to single-label instances. We show that `Pivot` remains a 3-approximation even in the multi-label setting. Further, we extend our heuristics to the new model and compare it with other heuristics (with their natural extensions to the multi-label setting) experimentally. We find that the algorithms compare with each other in a similar manner as in the single-label setting.

1.3 Related Work

The (non-chromatic) Correlation Clustering problem has been subject to a lot of research for over 20 years. It was first introduced by Ben-Dor et al. [6] for gene patterns. Bansal et al. [5] proved it to be NP-hard and gave the first constant-factor approximation. Charikar et al. [9] gave an improved approximation with ratio 4 and showed APX-hardness. Ailon et al. [2] proposed the linear time `Pivot` algorithm for CC and proved it to be a randomized 3-approximation. They also proposed a randomized 2.5-approximation with edge probabilities determined by an LP relaxation. Chawla et al. [10] gave an improved rounding strategy for the LP that gives a randomized 2.06-approximation. The LP-based approaches become infeasible for even moderate problem sizes as the number of constraints is cubic in the number of vertices.

There are many heuristics known for Correlation Clustering. Elsner and Schudy [15] compared three greedy algorithms `First`[22], `Best`[20], and `Vote`[14] and showed that `Vote` gives the best performance empirically. `Vote` iteratively chooses a random vertex and adds it to the cluster that fits best, if such a cluster exists, otherwise the vertex forms a new singleton cluster. They improved the result of `Vote` by using a post-processing local search `BOEM`, which repeatedly performs the best one element move to improve the clustering. However, this post-processing is very computation-intensive and can in practice only be used on small graphs. Lingas et al. [19] proposed an algorithm called Randomized Maximum Merging `RMM` that starts with singleton clusters and then iteratively merges the clusters that reduce the error the most (with random tiebreakers).

Chromatic Correlation Clustering was proposed by Bonchi et al. [7] who also gave a randomized degree-bound approximation called `CB` (Chromatic Balls). Here, a pivot edge is chosen uniformly at random and all vertices that form a monochromatic triangle with the pivot edge are added to its cluster. Anava et al. [4] proposed the first constant-factor approximation, a linear time 11-approx. `RC` (Reduce and Cluster) based on `Pivot` from Ailon et al. [2]. `RC` deletes

all secondary edges and runs Pivot on the resulting monochromatic components. They also proposed a randomized 4-approximation using linear programming. For better practical results they invented DC (Deep Cluster). It starts by deleting secondary edges as well, but after clustering the neighborhood of a pivot vertex it also considers the extended neighborhood (distance of two to the pivot vertex) and adds some of these vertices if that results in a lower error.

Recently, Amburg et al. [3] studied categorical edge clustering, a problem similar to CCC; their objective differed in that they do not penalize non-edge errors, the number of clusters is fixed, and only one cluster is allowed per color.

There is also a version of Correlation Clustering that maximizes agreements instead of minimizing disagreements. This variant admits a PTAS [5] as opposed to the APX-hard minimization variant. Also, there is the weighted variant of CC where edges have weights associated to them [5]. CC is further studied in a scenario where some vertex-pairs are not considered in the objective function; this models a practical scenario where pairwise relationship information between some pairs is unknown. This variant is harder and is unlikely to admit any constant approximation due to the equivalence with the Multi-Cut problem [13]. In this work, by CC we refer to the unweighted, minimizing disagreements version throughout.

2 RANDOMIZED 3-APPROXIMATION

The Pivot algorithm is a well-known heuristic for CC proposed by Ailon et al. [2]. By simply ignoring the colors, this algorithm can also be run on CCC instances; after the clusters are built in a color-blind fashion by the CC heuristic, one adds a color function that assigns to each cluster its most common color (see Algorithm 1). Bonchi et al. [7] used this algorithm as a baseline in their experiments for comparing CCC algorithms, although there was no theoretical guarantee known for Pivot in this chromatic setting. We show that Pivot is a 3-approximation for CCC, by a novel analysis.

Ailon et al. [2] show that Pivot is a 3-approximation in expectation for CC by charging the disagreements of the heuristic solution to *bad triples* (triples of vertices that induce a path of length two). When analyzing Pivot for the generalization to CCC, the addition of colors adds a new type of disagreement to be bounded: edges with wrong color inside a cluster. The problem when estimating these color-disagreements is that they depend on the unknown coloring function of an optimum solution.

In our analysis, we therefore fix an optimum solution with its coloring function and charge the disagreements of Pivot *directly* to the disagreements of this optimum solution. Note that this approach also yields an alternative proof for Pivot being a 3-approximation for CC. Our direct charging to the optimum has the advantage that we can estimate the ratio locally (in contrast to the previous global comparison via an LP relaxation for hitting bad triples) which might be a useful strategy also for other generalizations of CC.

Theorem 2.1. Pivot *is a 3-approximation in expectation for Chromatic Correlation Clustering.*

Proof. Let G be an edge-colored graph, $Opt = (C^*, ccol^*)$ be an optimal clustering, and $Sol = (C, ccol)$ be the result of Pivot on G. Let $Sol' = (C, ccol')$ where for each cluster $C_1 \in C$ with cluster-center p, $ccol'(C_1) = ccol^*(p)$. Note that by cluster-center, we mean the pivot whose neighborhood formed the cluster. Since Sol selected the best possible cluster colors, it is clear that $d(Sol) \leq d(Sol')$. Hence, it is sufficient to prove that $d(Sol') \leq 3 \cdot d(Opt)$.

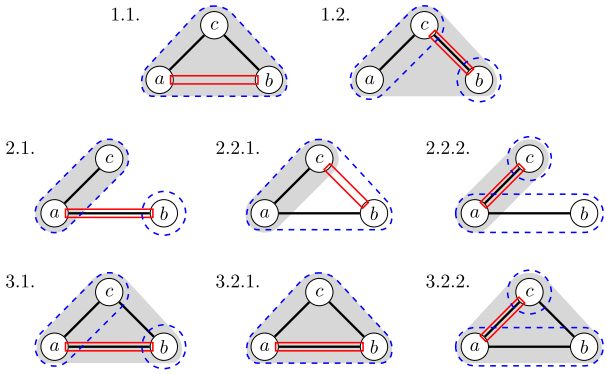

Figure 1: Charging cases in the proof of Theorem 2.1. Gray: Cluster of Sol' during critical iteration of ab. Blue: Cluster of Opt. Red: Charged edge. Note that slight variants of these examples are possible.

We charge each disagreement of Sol' to a disagreement of Opt such that in total each disagreement of Opt is charged at most 3 times in expectation, thus proving that Pivot is a 3-approximation.

We define the *critical iteration* of $ab \in \binom{V}{2}$ as the iteration at whose beginning a and b are not yet clustered and at whose end at least one of them is in a cluster. Hence, at the end of the critical iteration, it is determined whether ab is a disagreement in Sol'.

Let ab be a disagreement in Sol'. Without loss of generality, assume that a was clustered before or in the same iteration as b. Let c be the pivot picked in the iteration that clustered a (i.e. the critical iteration of ab). We charge a pair $B_{ab} \in \{ab, bc, ac\}$ for ab such that $B_{ab} \in D(Opt)$, as follows (see also Figure 1):

1. If ab is a non-edge within a cluster of Sol': then ac and bc are both edges as c is the cluster-center.
   1.1. If a and b are in the same cluster in Opt: then $ab \in D(Opt)$. We charge $B_{ab} := ab$.
   1.2. If a and b are in separate clusters in Opt: then either ac or bc is an edge between clusters in Opt and so in $D(Opt)$. We charge $B_{ab} := ac$ if $ac \in D(Opt)$, and $B_{ab} := bc$ otherwise.

Algorithm 1: Pivot

Data: An undirected, edge-colored Graph $G = (V, E, col)$
Result: A clustering $C = \{(C_1, c_1), \ldots (C_m, c_m)\}$ with $C_i \subseteq V$ and $c_i \in \mathbb{N}$.

Pick a random pivot $v \in V$ as cluster-center;
$C \leftarrow \{v\}$;
for $u \in N(v)$ **do**
  $C \leftarrow C \cup \{u\}$;
$c \leftarrow argmax_{c \in Colors} |\{ab \in E \cap C^2 \mid col(ab) = c\}|$;
return $\{(C, c)\} \cup Pivot(G[V \setminus C])$;

2. If ab is an edge between clusters of Sol': then ac is an edge and bc is a non-edge.
2.1. If a and b are in separate clusters of Opt: then $ab \in D(Opt)$. We charge $B_{ab} := ab$.
2.2. If a and b are in the same cluster of Opt.
2.2.1. If c is also in that cluster in Opt: then $bc \in D(Opt)$. We charge $B_{ab} := bc$.
2.2.2. If c is in a different cluster than a, b: then $ac \in D(Opt)$. We charge $B_{ab} := ac$.
3. If ab is an edge in a cluster of Sol' but does not have the color of its cluster:
3.1. If a and b are in separate clusters of Opt: then $ab \in D(Opt)$. We charge $B_{ab} := ab$.
3.2. If a and b are in the same cluster of Opt:
3.2.1. If c is in the same Opt cluster as a, b: then the Sol' cluster has the same color as the Opt cluster due to the cluster-color choice of Sol', so ab is also a wrong-color disagreement in Opt. We charge $B_{ab} := ab$.
3.2.2. If c is in a separate Opt cluster: then ac is a disagreement in Opt. We charge $B_{ab} := ac$.

For $e \in D(Opt)$, we denote by $M_e \subseteq D(Sol')$, the set of disagreements of Sol' charged to edge e during a run of `Pivot`.

Lemma 2.2. *For each disagreement $e \in D(Opt)$, the expected size of M_e is at most 3.*

Proof. Let uv be a disagreement in Opt. Consider the critical iteration of uv. Observe that vertex-pairs are only charged in their critical iteration. Let $N(u), N(v)$ be the neighborhoods of u and v respectively, excluding both u and v, at the beginning of the critical iteration of uv. Let $S = N(u) \cup N(v) \cup \{u, v\}$ and $s = |S|$. It is clear that a vertex from S is picked as cluster center, as at least one of u and v is clustered in this critical iteration. As the algorithm chooses a vertex uniformly at random in each iteration, all vertices in S have the probability $\frac{1}{s}$ of being picked. We upper bound the expected size of M_{uv} for the different types of disagreements.

- If uv is a non-edge in a cluster of Opt:
  As uv is a non edge, uv could be charged according to the construction above only in 1.1 or in 2.2.1. By construction, uv can be charged through 1.1 only for itself. Moreover, this happens only if a common neighbor from $N(u) \cap N(v)$ is picked as cluster center. Through 2.2.1 uv can be charged only if either u or v is picked as cluster center. If u is picked as cluster-center then $M_{uv} = \{vw \mid w \in N(u) \cap N(v)\}$ and if v is picked then $M_{uv} = \{uw \mid w \in N(u) \cap N(v)\}$. Thus,
$$\begin{aligned}\mathrm{E}[|M_{uv}|] &= \frac{|N(u) \cap N(v)|}{s} 1 + \frac{2}{s} |N(u) \cap N(v)| \\ &= \frac{3}{s}|N(u) \cap N(v)| < 3.\end{aligned}$$
- If uv is an edge between clusters of Opt:
  Here, there are 5 cases that could charge uv. The cases 2.1 and 3.1 only charge uv for itself and can occur only if neither u nor v is chosen as cluster center. Through 2.1, uv is charged only if a vertex of $N(u) \oplus N(v)$ (symmetric difference) is picked as cluster center. Through 3.1, uv is charged only if a vertex from $(N(u) \cap N(v))$
  The remaining 3 cases 1.2, 2.2.2 and 3.2.2 can occur only if either u or v is picked as the center. W.l.o.g. let the center be u. Through 1.2, uv can be charged only for the non-edges between v and $N(u) \setminus N(v)$. Through 2.2.2, uv can be charged only for the edges from v to $N(v) \setminus N(u)$. Through 3.2.2, uv can be charged only for the edges from v to $N(u) \cap N(v)$. Therefore,
$$\mathrm{E}[|M_{uv}|] \leq \tfrac{s-2}{s} 1 + \tfrac{2}{s}(s-2) < 3.$$
- If uv is an edge in a cluster of Opt with disagreeing color: Only 3.2.1 charges such disagreements, and there uv is only charged for itself. Hence, $\mathrm{E}[|M_{uv}|] \leq 1$.

Thus, in all cases, the expected size of M_e is at most 3. □

Since we have charged each disagreement of Sol' to a disagreement of Opt, and each disagreement of Opt was charged at most 3 times in expectation, we have that

$$\mathrm{E}[|D(Sol')|] \leq 3|D(Opt)|.$$ □

3 MONOCHROMATIC REDUCTION

Anava et al. [4] proposed a fast algorithmic framework to approximate CCC by using an approximation algorithm for the non-chromatic version of the problem. They delete all secondary edges E_2 to create the graph G_1 consisting of monochromatic components, which are then approximated by a (non-chromatic) CC algorithm.

Anava et al. use this framework in combination with `Pivot` and call it Reduce and Cluster (`RC`). They show that this yields an 11-approximation in expectation. In general, they show that given an α-approximation algorithm for CC, this technique yields a $(3 \cdot \alpha + 2)$ approximation. We improve their analysis and show the following:

Theorem 3.1. *The algorithmic framework by Anava et al. [4] when used with an α-approximation for Correlation Clustering, guarantees an $(\alpha + 2)$-approximation for Chromatic Correlation Clustering.*

Proof. First, we bound the number of secondary edges E_2. The following is implied by Lemma 2.3 in Anava et al. [4].

Lemma 3.2 (Secondary-Edge Lemma). $|E_2| \leq 2d^*(G)$.

Next, we argue that the optimum of G_1 (the graph after deleting secondary edges) is at most the optimum of G, improving on the bound by Anava et al. [4]. Towards this, we prove the following.

Lemma 3.3. *Given a graph G, a vertex v and a color c. If G' is the graph derived from G by removing all edges of color c incident to v, then $d^*(G') \leq d^*(G)$.*

Proof. Let Opt be an optimum clustering on G. If the cluster color of v in Opt is different from c, then $d(G', Opt) \leq d(G, Opt)$ as deleting the edges only removes inter-cluster disagreements and turns wrong-color into non-edge disagreements. If the cluster of v has color c, we define Opt_v as Opt but with v in its own cluster. We see that $d(G', Opt_v) \leq d(G, Opt)$ as all disagreements incident to v on G' were either color or inter-cluster disagreements before. In both cases there exists a clustering for G' that does not have more disagreements than $d^*(G)$, consequently $d^*(G') \leq d^*(G)$. □

Now, let C be a clustering produced by an α-approximate CC algorithm on the (monochromatic) connected components of G_1. Since the components are CC instances, we have $d(C, G_1) \leq \alpha d^*(G_1)$. By

Lemma 3.3, we know that $d(C, G_1) \leq \alpha d^*(G)$. Interpreting C as a clustering on G can, in the worst-case, result in all secondary edges being additional disagreements, thus $d(C, G) \leq d(C, G_1) + |E_2| \leq \alpha d^*(G) + 2d^*(G)$, where the last inequality is by Lemma 3.2. Thus we conclude that the framework gives a $(\alpha + 2)$-approximation. □

For RC in particular, the theorem implies:

Corollary 3.4. RC *is a 5-approximation in expectation for Chromatic Correlation Clustering.*

4 GREEDY EXPANSION

While Pivot and RC give constant factor approximations in theory, heuristics without proven guarantees like CB (Bonchi et al. [7]) and DC (Anava et al. [4]) outperform them on practical instances, as shown by experiments in the respective publications. DC in particular starts building a cluster like Pivot but then expands it further by greedily adding vertices of distance two from the cluster center, as long as it is deemed beneficial. The idea of incrementally adding vertices to clusters can also be found in the literature for CC. Lingas et al. [19] describe that RMM starts with singleton clusters and then iteratively merges the two clusters that give the highest error reduction. However, it can be observed that RMM often builds one cluster at a time before moving on to the next. This can be explained by observing that merging two connected singleton clusters always has a gain of one, whereas the gain of merging a singleton into a larger cluster can be much higher. RMM can be extended to the CCC setting by accounting for colors when calculating the merge gain. However, in preliminary experiments we noticed that this tends to make bad choices for the first edges to merge.

Algorithm 2: Greedy Expansion

Data: An undirected, edge-colored Graph $G = (V, E, col)$
Result: A clustering $Sol = \{(C_1, c_1), \dots (C_m, c_m)\}$ with $C_i \subseteq V$ and $c_i \in \mathbb{N}$.

Algorithm greedy_expansion(*G*,*g*=*25*)
  $Sol \leftarrow \emptyset$;
  while $E \neq \emptyset$ **do**
    choose $uv \in E$ u.a.r.;
    $(C, c) \leftarrow$ expand_around(*u*,*v*,*g*);
    $Sol \leftarrow Sol \cup \{(C, c)\}$;
    $V \leftarrow V \setminus C$; $E \leftarrow E \setminus \{uv \in E \mid u \in C\}$;
  Make remaining vertices in V singleton clusters;
  return *Sol*;
Function error(*C*)
  return number of errors in C when colored optimally;
Function merge_gain(*C*,*w*)
  return error(*C*) + $|E \cap (C \times \{w\})|$ − error($C \cup \{w\}$);
Procedure expand_around(*u*,*v*, *g*)
  $C \leftarrow \{u, v\}$;
  Draw up to g vertices $w \in N(u) \cup N(v)$ and add them to C, prefer vertices is $N(u) \cap N(v)$;
  while $\min_{w \in C}$ merge_gain($C \setminus \{w\}, w$) ≤ 0 **do**
    $w \leftarrow arg\min_{w \in C}$ merge_gain($C \setminus \{w\}, w$) ;
    $C \leftarrow C \setminus \{w\}$;
  while $\max_{w \in V \setminus C}$ merge_gain(C, w) > 0 **do**
    $w \leftarrow arg\max_{w \in V \setminus C}$ merge_gain(C, w) ;
    $C \leftarrow C \cup \{w\}$;
  $c \leftarrow$ best color for C ;
  return (C, c);

As the first vertices added to a cluster have a high impact on its future growth, we propose our own heuristic that focuses more on the initial stage of a cluster. The idea is to stabilize the clusters at their initial stage by exploring potential clusters in the immediate neighbourhood before greedily adding additional vertices. Our heuristic Greedy Expansion (GE) iteratively chooses an unclustered edge and builds a cluster around it as described in algorithm 2. First, we consider the neighborhood of the edge and temporarily add a sample of it to our cluster. Then, vertices that do not fit are iteratively removed to produce a good starting cluster. As taking the whole neighborhood can be time consuming for vertices with high degree, we restrict the sample size of neighbored vertices with the parameter g. To increase the chances of finding a good initial cluster, we prefer vertices from the common neighborhood of the starting edge for our sample. Then, the initial cluster is greedily expanded with the vertices that maximise the merge gain.

GE can be implemented to run in $O(|V|\Delta + |E|\Delta_L)$ for constant g (we chose $g = 25$ as this resulted in a good trade-off between quality and running time in our experiments), where Δ is the maximum degree of G and Δ_L the maximum number of distinct colors incident to a vertex. (See Appendix A for a formal proof.) This worst-case runtime seems to be overly pessimistic and the algorithm runs much faster in the real-world datasets as can be seen in Table 3.

5 EXPERIMENTAL EVALUATION

To evaluate the practical effectiveness of our algorithms, we compare them with the state-of-the-art for CCC: Chromatic Balls (CB) from Bonchi et al. [7], Reduce and Cluster (RC) and Deep Cluster (DC) from Anava et al. [4], on synthetic and real-world data. While their experiments have already shown RC to be inferior, we include it to compare with Pivot, which offers the better theoretical guarantee.

Both RC and DC start by deleting all secondary edges, leaving only monochromatic subcomponents. We therefore also consider known CC heuristics with this preprocessing step. We evaluate the performance of Vote from Elsner and Charniak [14] and Random Maximum Merging (RMM) from Lingas et al. [19] on the graph without secondary edges. For our proposed heuristic GE, we use an initial growth limit of $g = 25$. Furthermore, we include a modified version GER which runs on the preprocessed graph. In GER, the difference is that we do not consider colors when calculating the error, as GER runs on graphs with monochromatic components.

All our experiments were conducted on an Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz with 64GB of RAM. The algorithms are implemented in Python 3 using networkx 2.5 [16]. To get access to the code and datasets, see Appendix B.

5.1 Synthetic

The synthetic data is generated identically to Anava et al. [4]. The graph created has n vertices and $|L|$ colors. Initially, each vertex is

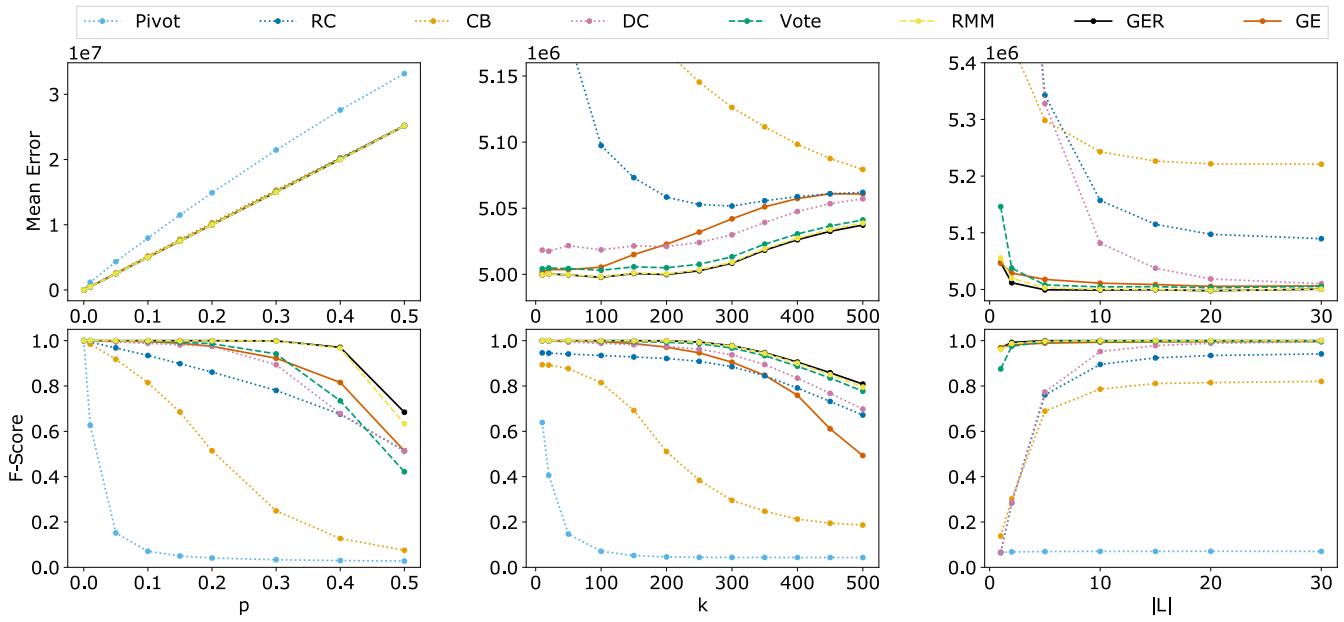

Figure 2: Experimental results on synthetic data. Mean error (number of disagreements) in upper, and F-Score (1.0 for perfect reconstruction of the *ground truth*) in lower row, averaged over 10 graphs with 5 runs on each. One parameter of $p, k, |L|$ is varied at a time keeping the others at $n = 10000$, $p = 0.1$, $k = 100$ and $|L| = 20$. `Pivot` is omitted if it is far out of the plotted area.

assigned u.a.r. to one of k clusters. Each cluster has a color drawn u.a.r. and all vertices within it are connected with an edge of the cluster color. This cluster graph is then modified by random noise. We change each $uv \in \binom{V}{2}$ with a probability p. If $uv \notin E$, we create a new edge with a color drawn u.a.r. If $uv \in E$, we draw a color u.a.r. and recolor the edge or remove it if we draw its current color. We use the initial assignment of vertices to the k clusters as *ground truth*, although it is not guaranteed to be an optimum solution.

As default parameters, we use $n = 10000$, $p = 0.1$, $k = 100$ and $|L| = 20$. We evaluate the impact of one parameter at a time on the algorithms' performances. For each parameter combination, we measure the mean over 10 graphs with 5 runs on each per algorithm.

We measure the number of disagreements and the F-score for clustering, which reflects the similarity between a clustering and the *ground truth*. For a ground truth $(C_i^*, c_i^*)_{i\in[k^*]}$ and a clustering $(C_i, c_i)_{i\in[k]}$, it is computed as follows

$$F(C, C^*) = \frac{1}{n} \sum_{i=1}^{k^*} |C_i^*| \max_{1\le j\le k} F_{ij},$$

where $F_{ij} = (2P_{ij}R_{ij})/(P_{ij} + R_{ij})$ and $R_{ij} = |C_i^* \cap C_j|/|C_i^*|$ is recall and $P_{ij} = |C_i^* \cap C_j|/|C_j|$ is precision.

Figure 2 shows the results. As expected, increasing the noise p increases the error and makes it harder to find the original clustering. When the number of colors $|L|$ is increased, it is easier to distinguish noise edges, as they are less likely to have the cluster color. Increasing k decreases the expected cluster size which makes it more difficult to differentiate them from noise.

In all settings, we clearly see that `Pivot` is outperformed by all other algorithms, despite having the best proven approximation guarantee. We also see that the adapted CC heuristics `RMM` and `Vote` perform much better than the previous heuristics suggested for CCC (`CB`, `DC`). Further, we observe that `GE` is outperformed by `GER`, which together with `RMM` yields the best results by a slim margin. Intuitively, this might be the effect of removing secondary edges which on these synthetic graphs reduces the noise and makes clustering easier. We see that in all but the most extreme parameter settings, the CC algorithm `RMM` and especially our heuristic `GER` are able to reconstruct the *ground truth* almost perfectly.

Dataset	|V|	|E|	|L|	p_μ
Facebook	2,884	62,334	193	8.3%
DAWN	2,109	96,047	10	41.4%
MAG	80,198	237,261	11	7.2%
String_S	18,152	401,582	4	n/a
Twitter	22,964	431,329	4,065	10.4%
Cooking	6,714	479,921	20	29.4%
DBLP_S	73,624	835,414	100	n/a
DBLP	2,578,154	14,470,369	6,606	25%
String	492,199	22,151,767	7	54.9%

Table 1: Overview of the real-world datasets used in the experimental section where p_μ is the percentage of edges with multiple colors in the multi-chromatic version in section 6.

5.2 Real-World Data

We evaluate the algorithms on real-world edge-labeled data from various domains. Table 1 gives an overview of our datasets.

Dataset	Mean Error								Cluster Count							
	Pivot [2]	RC [4]	DC [4]	CB [7]	Vote [14]	RMM [19]	GER [ours]	GE [ours]	Pivot [2]	RC [4]	DC [4]	CB [7]	Vote [14]	RMM [19]	GER [ours]	GE [ours]
Facebook	56,245	0.881	0.829	0.853	0.722	0.703	0.705	**0.700**	560	1.430	1.405	1.655	1.682	1.645	1.618	1.355
DAWN	114,636	0.911	0.884	0.824	0.783	0.765	0.763	**0.734**	983	1.149	1.151	1.249	1.318	1.339	1.323	1.209
MAG	199,325	0.823	0.824	0.689	0.677	0.679	0.665	**0.653**	29,927	1.067	1.061	1.140	1.151	1.145	1.144	1.115
String_S	161,469	0.970	0.848	0.994	0.748	**0.703**	0.708	0.708	1,038	1.075	0.837	1.397	1.114	1.061	0.991	0.952
Twitter	572,772	0.702	0.693	0.699	0.674	0.664	0.662	**0.658**	6,267	1.528	1.479	1.651	1.672	1.628	1.623	1.373
Cooking	712,742	0.702	0.706	0.665	0.645	0.641	0.641	0.643	2,775	1.208	1.205	1.271	1.386	1.405	1.391	1.226
DBLP_S	838,504	0.736	0.711	0.736	0.690	0.674	0.670	**0.654**	9,392	2.448	2.282	2.467	2.627	2.496	2.481	1.556
DBLP	16,534,591	0.703	0.701	0.666	0.660	0.659	0.656	**0.642**	1,040,069	1.239	1.230	1.204	1.289	1.282	1.281	1.159
String	28,076,379	0.852	0.838	0.770	0.731	0.728	0.726	**0.705**	213,982	1.083	0.993	1.226	1.184	1.174	1.096	1.010

Table 2: Experimental results on real-world datasets. Mean error and cluster count normalized with Pivot, i.e. divided by the result of Pivot, given in the Pivot columns. Lowest errors are highlighted in bold, if the difference is statistically significant.

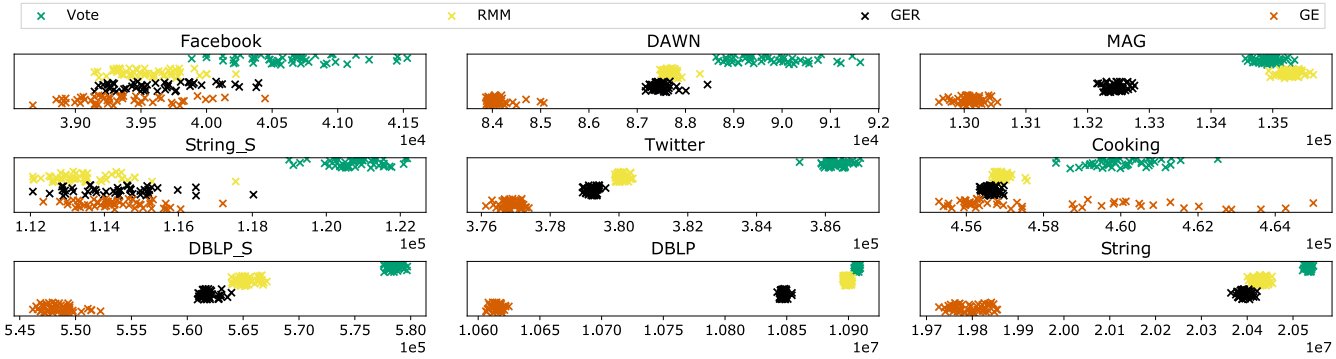

Figure 3: The scatter plots display the distribution of the number of disagreements across the 50 runs on single-label data. It is restricted to the best performing heuristics for scaling purposes.

Facebook and Twitter are social network graphs from [18]. Here, vertices represent persons which are connected if they are friends (or one follows the other). Similarly to the Facebook and Twitter datasets of Anava et al. [4], we colored edges based on the social circles attributes of the dataset. When two vertices are grouped within the same social circle, the edge between them is labeled with that circle; when they belong to multiple circles we choose one of them randomly for the single-label dataset.

The DBLP_S and String_S datasets are identical to those used in [4, 7]. Since these are rather small and only fit the single-label setting, we created our own version of these from the same sources.

DBLP is a coauthorship network obtained from DBLP[1]. Vertices (authors) are connected with an edge labeled with the journal or conference they published in together most frequently. The Microsoft Academic Graph (MAG) obtained from [3, 21] also categorizes co-author relationships based on their most common venue.

STRING DB [23] is a database on protein-protein interactions. In particular, the table 'network' contains pairs of interacting proteins along with a mode of interaction. This can be modeled as a graph String where proteins are represented by vertices, interactions by edges, and the modes of interaction by edge-colors. For edges with multiple types of interaction, one of them is chosen u.a.r. for the single-label data. As the whole String DB is very large, we only used a subset of it. As the graph consists of multiple components, we evaluate them separately to reduce the memory consumption.

Further, we include two datasets used by Amburg et al. [3] who studied a different model of clustering with categorical edge-labels. We select the two datasets with highest percentage of multi-label-edges aiming for interesting graphs for our setting in section 6. The DAWN dataset [1] models a drug abuse warning network. Vertices are drugs and edges mean that they were used together before an emergency room visit. The color represents the most common outcome of that visit. In the Cooking dataset [17], vertices are ingredients that are connected with an edge if they co-occur in recipes with a color of the most common cuisine of those.

On each dataset, we ran the algorithms 50 times each and averaged the error. To account for the randomness of the experiments, we performed a Mann-Whitney-U Test to test for statistical significance with a p threshold of 0.01. The results are given in Table 2. As before, we see that Pivot gives the worst results. Deleting secondary edges beforehand (RC) already gives a drastic improvement. The heuristics suggested by previous papers on CCC generally perform better, but there is no clear winner among CB and DC. Both of them are significantly outperformed by the CC algorithms and our heuristics on all datasets. Of those, Vote loses to RMM on all but one datasets. Comparing GE to the best of previous CCC heuristics,

[1]The dblp team: dblp computer science bibliography. Monthly snapshot release of December 2020. dblp.org

Dataset	Pivot	RC	DC	CB	Vote	RMM	GER	GE
Facebook	0.04	0.35	0.54	0.06	0.34	1.42	0.65	0.77
DAWN	0.04	0.44	0.90	0.10	0.46	1.71	0.85	1.18
MAG	0.43	2.16	2.43	0.61	2.36	6.46	3.99	3.68
String_S	0.47	2.65	2.98	0.56	2.41	11.36	4.57	2.57
Twitter	0.29	2.10	2.35	0.79	2.15	4.85	2.90	14.64
Cooking	0.19	1.93	3.02	0.69	2.00	5.43	2.96	7.57
DBLP_S	0.91	6.07	6.27	2.20	6.09	16.20	8.77	18.24
DBLP	24.29	122.90	132.50	52.20	130.57	310.48	176.34	338.26
String	12.28	102.92	258.67	33.36	108.75	417.47	206.19	287.33

Table 3: Wall clock times of the algorithms on the real-world data in seconds. The values are averaged over 50 runs.

the largest improvement is 16.5% on String_S, and the smallest is 3.4% on Cooking. GE significantly improves over RMM in the range 4.1% to 2.6% on five datasets. In contrast to the synthetic data, here GE outperforms its modification GER, which is more similar to RMM. On the two datasets where GE is not the best, the difference to the best one is only 0.7% and 0.3%. On all but two datasets, the statistical testing shows that GE is significantly better than all other algorithms. Looking further into the measurements, we see that on six datasets GE's worst run is better than the best run of any other algorithm (see Figure 3). On String_S, RMM has a statistically significant lead, yet the difference in the distributions is less clear.

The run times of the algorithms are presented in Table 3. GE and RMM are generally the slowest, yet fast enough considering the size of the input data; e.g., the largest time taken by GE is about 5 minutes on the DBLP dataset that has about 2.5 million vertices and 15 million edges. We remark that we did not optimize the code for run time, as all runtimes are well under feasible times.

In Table 2, we have also included the mean number of clusters found. It can be seen that GE builds relatively few clusters compared to other algorithms except Pivot and DC.

6 A NOVEL MULTI-LABEL MODEL FOR CCC

Motivated by the real-world datasets of CCC, we propose a new multiple edge-label model of CCC. Our proposition is based on the following limitation of CCC we observed. Often, edges in the datasets have multiple potential labels, i.e. in a social network two people A and B might be schoolmates as well as colleagues, and two proteins in a protein-protein interaction network often have multiple types of interaction. When using this data in the single-label clustering setting, it has to be preprocessed to only have one of these colors, for example by choosing the most prominent label. But this causes errors in the model that do not reflect the real-world scenario. Consider our social network example where A and B are preprocessed to have a *colleague-edge* between them. It could then happen that they are clustered into a *schoolmates cluster* (possibly due to having many common schoolmates), then the edge AB counts as an error even though they are in fact schoolmates.

To counter this limitation of the CCC model, we propose the *Single-Satisfy Multi-Chromatic Correlation Clustering (SSMCCC)* model. Here, each edge has a set of accepted colors $Col\colon E \to 2^L$ associated with it. The solutions assign a single color to each cluster as in CCC. An edge uv counts as a wrong-color disagreement if $C(u) = C(v)$ and $ccol(C(u)) \notin Col(uv)$. The edges across clusters and non-edges within clusters remain disagreements.

We remark that Bonchi et al. [7] also considered a different multi-label model. They assign to clusters not just one but a set of colors and the color errors are calculated as the Hamming distance between cluster and edge colors. The issue of not properly reflecting real-world data can occur even in this model. Thus, we opted to study our new multi-label model, hoping to resolve this.

We refer to the number of disagreements of a solution in the SSMCCC model as $d_M(G, Sol)$ and to the minimum number of disagreements as $d^*_M(G)$. Observe that the cost of a clustering will only improve, when allowing additional colors for an edge. More formally the following lemma holds.

Lemma 6.1. *Given a graph G and a clustering Sol. Let G' be derived from G by an additional accepted color at an already existing edge. Then $d_M(G', Sol) \leq d_M(G, Sol)$.*

Using this lemma, we show that Pivot also retains its approximation guarantee in the single-satisfy setting.

Theorem 6.2. Pivot *is a 3-approximation in expectation for Single-Satisfy Multi-Chromatic Correlation Clustering.*

Proof. Let G be an instance of SSMCCC and Opt a respective optimum solution. We define a CCC instance G' depending on G and Opt as follows. G' has the same set of edges as G, and the color of uv in G' is assigned as follows: if uv is an agreement in Opt, then assign uv the cluster color of its cluster in Opt, otherwise assign it an arbitrary color from its set of acceptable colors in G. Then by Lemma 6.1 and the construction of G' we see that

$$d^*(G') = d(G', Opt) = d_M(G, Opt) = d^*_M(G).$$

As Pivot is an expected 3-approximation for CCC, we know that

$$\mathrm{E}[d(G', Pivot(G'))] \leq 3d^*(G').$$

Since Pivot does not look at colors for the clustering part, it outputs a specific partition of the vertex-set on G, with the same probability as it would have on G'. Since each cluster color is determined by majority vote, the associated clustering produces no more errors on G than on G'. Hence, the expected error of Pivot on G is at most the expected error on G'. Thus, we get:

$$\begin{aligned}\mathrm{E}[d_M(G, \texttt{Pivot}(G))] &\leq \mathrm{E}[d(G', \texttt{Pivot}(G'))] \\ &\leq 3d^*(G') \leq 3d^*_M(G).\end{aligned}$$ □

While Pivot retains its approximation guarantee for SSMCCC, this is not the case for the natural generalization of RC. We can create monochromatic components by interpreting the instance as a multigraph (where each edge has exactly one color) and then remove all secondary edges. The following example shows why the error caused by this is no longer bounded by a constant factor.

Let $n \in \mathbb{N}$ and $L = [n]$. Consider a graph with vertices $\{a_i\}_{i\in[n]} \cup \{b_i\}_{i\in[n]}$. For all $i, j \in [n], i \neq j$, there is an edge of all colors from a_i to a_j, so $\{a_i\}_{i\in[n]}$ is a clique. For all $i \in [n]$, there is an edge of color i between a_i and b_i. Evidently, the optimum solution clusters $\{a_i\}_{i\in[n]}$ into one big cluster and isolates all vertices from $\{b_i\}_{i\in[n]}$, thus incurring n disagreements. On the other hand, all vertices from $\{a_i\}_{i\in[n]}$ have distinct primary colors, so deleting secondary edges removes all edges from the clique and leads to a pairwise clustering between a_i and b_i with an error in $O(n^2)$.

Dataset	Pivot [2]	RC [4]	DC [4]	Vote [14]	RMM [19]	GER [ours]	GE [ours]
Facebook	54,978	0.886	0.843	0.720	0.697	0.697	0.695
DAWN	106,678	0.936	0.962	0.776	0.764	0.763	**0.708**
MAG	196,397	0.833	0.835	0.680	0.683	0.668	**0.656**
Twitter	537,268	0.730	0.719	0.695	0.684	0.683	**0.681**
Cooking	612,092	0.813	0.808	0.746	0.741	0.741	**0.736**
DBLP	16,595,563	0.696	0.696	0.653	0.654	0.651	**0.630**
String	26,020,637	0.900	0.868	0.688	0.694	0.694	**0.660**

Table 4: Experimental results on real-world multi-label data normalized by Pivot. Lowest errors in bold, if the difference is statistically significant.

To evaluate our suggested multi-label setting, we use the same datasets as described in subsection 5.2, except that we allow multiple labels for edges instead of enforcing single labels.

Our heuristic can naturally be extended to the multi-label setting by creating copies of each edge uv for each of its accepted colors, and then working with the resultant multi-graph. Since most of the other algorithms delete secondary edges and work with monochromatic components, they can also be generalized. Note that we did not run CB here, as it has no clear generalization to the multi-label scenario. We again run each algorithm 50 times on each dataset and use a U-Test for statistical significance as in subsection 5.2.

Looking at the results in Table 4, we observe that the algorithms that delete secondary edges, although proven to be bad on some SSMCCC instances, still perform much better than the theoretically proven 3-approximation Pivot. As before, the adapted CC heuristics Vote and RMM are much better than DC. On all datasets but Facebook, statistical analysis proves that GE performs significantly better than the other algorithms. On four of them, the improvement over RMM in mean error ranges from 3.6% to 7.3%.

Comparing Table 4 and Table 2, we see a significant drop of errors for SSMCCC compared to CCC. Especially, GE on DAWN and String produces 10.2% and 13.3% less errors than on the single-label model. This supports our hypothesis that errors that do not model the real-world scenario can occur in the single-label setting.

The theoretical runtime of GE for SSMCCC remains the same as that of the single-label setting interpreting $|E|$ as the total number of labels across all edges. The experimental runtimes of all algorithms in the multi-label setting are more or less similar to those in the single-label setting, so we do not show them here.

7 FURTHER WORK

From a theoretical standpoint, it would be interesting to see if CCC admits better than 3-approximation in polynomial time. Looking at successful approaches for CC, it might be worth to reconsider LP relaxation. It is probably much harder to improve upon a ratio of 3 in linear time as this remains an open problem also for the more intensely studied CC. From a practical viewpoint, we think that our SSMCCC model offers a more realistic setting, and hence developing better heuristics in this model is an interesting direction.

ACKNOWLEDGMENTS

We thank Noa Avigdor-Elgrabli, for sharing the DBLP_S and String_S datasets with us.

REFERENCES

[1] Substance Abuse and Mental Health Services Administration. 2013. Drug Abuse Warning Network, 2011: National Estimates of Drug-Related Emergency Department Visits. *HHS publication no.(SMA)* 13 (2013), 4760.

[2] Nir Ailon, Moses Charikar, and Alantha Newman. 2008. Aggregating inconsistent information: Ranking and clustering. *J. ACM* 55, 5 (2008), 23:1–23:27.

[3] Ilya Amburg, Nate Veldt, and Austin R. Benson. 2020. Clustering in graphs and hypergraphs with categorical edge labels. In *WWW '20: The Web Conference 2020*. ACM / IW3C2, 706–717.

[4] Yael Anava, Noa Avigdor-Elgrabli, and Iftah Gamzu. 2015. Improved Theoretical and Practical Guarantees for Chromatic Correlation Clustering. In *Proceedings of the 24th International Conference on World Wide Web (WWW)*. ACM, 55–65.

[5] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. 2004. Correlation Clustering. *Mach. Learn.* 56, 1-3 (2004), 89–113.

[6] Amir Ben-Dor, Ron Shamir, and Zohar Yakhini. 1999. Clustering Gene Expression Patterns. *J. Comput. Biol.* 6, 3/4 (1999), 281–297.

[7] Francesco Bonchi, Aristides Gionis, Francesco Gullo, Charalampos E. Tsourakakis, and Antti Ukkonen. 2015. Chromatic Correlation Clustering. *ACM Trans. Knowl. Discov. Data* 9, 4 (2015), 34:1–34:24.

[8] Stefani Chan, Raymond K Pon, and Alfonso F Cárdenas. 2006. Visualization and clustering of author social networks. In *Distributed Multimedia Systems Conference*. 174–180.

[9] Moses Charikar, Venkatesan Guruswami, and Anthony Wirth. 2005. Clustering with qualitative information. *J. Comput. Syst. Sci.* 71, 3 (2005), 360–383.

[10] Shuchi Chawla, Konstantin Makarychev, Tselil Schramm, and Grigory Yaroslavtsev. 2015. Near Optimal LP Rounding Algorithm for Correlation Clustering on Complete and Complete k-partite Graphs. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, (STOC)*. ACM, 219–228.

[11] Flavio Chierichetti, Nilesh N. Dalvi, and Ravi Kumar. 2014. Correlation clustering in MapReduce. In *The 20th ACM International Conference on Knowledge Discovery and Data Mining, (SIGKDD)*. ACM, 641–650.

[12] Manlio De Domenico, Vincenzo Nicosia, Alexandre Arenas, and Vito Latora. 2015. Structural reducibility of multilayer networks. *Nature communications* 6, 1 (2015), 1–9.

[13] Erik D. Demaine, Dotan Emanuel, Amos Fiat, and Nicole Immorlica. 2006. Correlation clustering in general weighted graphs. *Theor. Comput. Sci.* 361, 2-3 (2006), 172–187.

[14] Micha Elsner and Eugene Charniak. 2008. You Talking to Me? A Corpus and Algorithm for Conversation Disentanglement. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*. The Association for Computer Linguistics, 834–842.

[15] Micha Elsner and Warren Schudy. 2009. Bounding and Comparing Methods for Correlation Clustering Beyond ILP. In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*. 19–27.

[16] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. 2008. Exploring Network Structure, Dynamics, and Function using NetworkX. In *Proceedings of the 7th Python in Science Conference*. 11 – 15.

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

[18] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data.

[19] Andrzej Lingas, Mia Persson, and Dzmitry Sledneu. 2014. Iterative merging heuristics for correlation clustering. *Int. J. Metaheuristics* 3, 2 (2014), 105–117.

[20] Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. 104–111.

[21] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June Paul Hsu, and Kuansan Wang. 2015. An Overview of Microsoft Academic Service (MAS) and Applications. In *Companion Proceedings of the 24th International Conference on World Wide Web (WWW)*. ACM, 243–246.

[22] Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational linguistics* 27, 4 (2001), 521–544.

[23] Damian Szklarczyk, Annika L. Gable, David Lyon, Alexander Junge, Stefan Wyder, Jaime Huerta-Cepas, Milan Simonovic, Nadezhda T. Doncheva, John H. Morris, Peer Bork, Lars Juhl Jensen, and Christian von Mering. 2019. STRING v11: protein-protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. *Nucleic Acids Res.* 47, Database-Issue (2019), D607–D613. https://doi.org/10.1093/nar/gky1131

A GREEDY EXPANSION: RUN TIME ANALYSIS

In this section, we analyze the run time of our GE heuristic and discuss how it could be implemented most efficiently. The analysis is theoretical though and does not reflect the implementation used during the experiments, as in practice, the algorithm runs efficiently even without some of those optimizations. In particular, we found it to be sufficient to do a linear scan for the best vertex to add, instead of using (bucket) priority queues.

We use Δ to refer to the maximum degree of the input graph and Δ_L for the maximum number of colors incident to a vertex. The graph is stored as an adjacency list. We focus on the function `expand_around` in algorithm 2, which is called at most $O(|V|)$ times.

When sampling from the neighborhood in line 16, we create an array of common neighbors and an array of exclusive neighbors by iterating through the adjacency lists of u and v and marking the respective vertices. Afterwards, we sample up to g vertices from those arrays, so the complexity is in $O(\Delta)$. Next, we copy the subgraph induced by the vertices of C in $O(g\Delta)$.

While shrinking the initial cluster during the loop in line 17, we work on this copied subgraph. We keep track of the number of edges of each color within the cluster, and incident to each vertex. For each of the $O(g)$ iterations, we check for each of the $O(g)$ vertices the removal gain in $O(\Delta_L)$ time based on this information. When a vertex gets removed, we must update its neighbors within the cluster. Overall this reqiures time in $O(g^2\Delta_L)$. Accumulated over the whole run of the algorithm, this gives us $O(|V|(g\Delta + g^2\Delta_L))$ time for the shrinking part, i.e. $O(|V|\Delta)$ for constant g.

We now analyse the expansion (loop in line 20) amortized across all calls of `expand_around`. Again, we keep track of the number of edges for each color within the cluster. For each vertex w neighbored to the cluster and each color $c \in L$, we also keep track of the number of edges $a_w^c = |\{wx \in E \mid x \in C \wedge col(wx) = c\}|$. To efficiently determine the best vertex to add, we maintain a priority queue for each color occurring. For color $c \in L$, the priority of a vertex w in the respective queue is computed as $a_w^c + \frac{1}{2}\sum_{c' \neq c} a_w^{c'}$. Moreover, we use a priority queue containing the top entries of each color queue adjusted for the respective number of colored edges within the cluster. The different edge counters are only updated when a vertex is added to the cluster permanently, so we can upper bound the number of updates with $O(|E|)$. When a vertex gets updated because a neighbor has been added to C, it must update its entries in all the priority queues it is part of, so in at most Δ_L queues. Since the priorities change in steps of one or one-half, we can perform them in constant time by using bucket queues. This allows for the expansion process to run in $O(|E|\Delta_L)$.

Overall, this yields a run time of $O(|V|\Delta + |E|\Delta_L)$ for constant g.

B DATA AND CODE

The code and data used in our experiments can be found in the following GitHub repository. Please consult the README for instructions on how to use the code. https://github.com/arthurz0/3-approx-for-ccc-and-heuristics