Fair Correlation Clustering in Forests

Fair Correlation Clustering in Wäldern

Simon Wietheger

Universitätsmasterarbeit
zur Erlangung des akademischen Grades

Master of Science
(M. Sc.)

im Studiengang
IT-Systems Engineering

eingereicht am 28. September 2022 am
Fachgebiet Algorithm Engineering der
Digital-Engineering-Fakultät
der Universität Potsdam

Gutachter	Prof. Dr. Tobias Friedrich Prof. Dr. Mathias Weske
Betreuer	Dr. Katrin Casel Dr. Martin Schirneck

Abstract

The study of fair algorithms receives growing attention as the awareness of bias in the input data for machine learning applications that results in discriminatory output increases. Fair Correlation Clustering is the fair variant of the well-known Correlation Clustering objective. Given a graph with colored vertices, it partitions the vertices fairly in the sense that the color distribution in each cluster matches the color distribution in the overall graph. The task is to obtain such a clustering with a minimum number of disagreements, i.e., pairs of adjacent vertices that are not in the same cluster and pairs of vertices that are not adjacent but in the same cluster.

While first results on the NP-hard Fair Correlation Clustering problem give various approximation algorithms, we focus on exact solutions and investigate whether there are efficiently solvable instances. As unfair Correlation Clustering is easily solved in forests, we study Fair Correlation Clustering in various kinds of forests. We find that Fair Correlation Clustering turns NP-hard very quickly, even when assuming constant degree or diameter of the forest. Nevertheless, we provide algorithms that have polynomial running time for certain assumptions on the color distribution in the graph.

Further, we analyze a relaxed fairness setting, where the color distribution of the clusters does not have to match the global one precisely. Instead, for each color, there is an upper and lower bound on the ratio of vertices of that color in each cluster. We prove that our algorithmic and hardness results essentially still hold in this relaxed setting and thereby show that the hardness of the problem is not due to the strict fairness definition. For the setting of exact fairness on forests, we give a PTAS that contrasts the APX-hardness of Correlation Clustering on general graphs. Additionally, we show that key insights on Fair Correlation Clustering in forests even hold in bipartite graphs and thereby hope to pave the way for future algorithmic results on more general graph classes.

Zusammenfassung

Faire Algorithmen erhalten durch das wachsende Bewusstsein für Bias in den Eingabedaten des maschinellen Lernens und sich daraus ergebenden diskriminierenden Resultaten immer mehr Bedeutung. Diese Arbeit behandelt Fair Correlation Clustering, welches die faire Variante des bekannten Correlation Clustering darstellt. Hierbei soll bei einem Graphen mit gefärbten Knoten die Knotenmenge fair zerlegt werden, so dass die Farbverteilung in jedem Cluster mit der Farbverteilung im Gesamtgraphen übereinstimmt. Die erhaltene Zerlegung soll dabei möglichst wenig Unstimmigkeiten enthalten, d. h. Paare von benachbarten Knoten, die nicht im selben Cluster sind, und Paare von Knoten, die nicht benachbart, aber im selben Cluster sind, sollen vermieden werden.

Während erste Forschungsergebnisse zu dem NP-schweren Fair Correlation Clustering verschiedene Approximationsalgorithmen entwickeln, konzentrieren wir uns in dieser Arbeit auf exakte Lösungen und untersuchen, ob es Instanzen gibt, die effizient lösbar sind. Da unfaires Correlation Clustering in Wäldern leicht zu lösen ist, untersuchen wir hier Fair Correlation Clustering in verschiedenen Arten von Wäldern. Wir stellen fest, dass Fair Correlation Clustering selbst bei Wäldern mit konstantem Grad oder Durchmesser NP-schwer ist. Allerdings entwickeln wir Algorithmen, die für bestimmte Annahmen über die Farbverteilung im Graphen polynomielle Laufzeiten haben.

Darüber hinaus untersuchen wir eine weiter gefasste Fairness-Definition. Nach dieser muss die Farbverteilung der Cluster nicht genau mit der globalen Verteilung übereinstimmen, sondern es reicht aus wenn der Anteil jeder Farbe in jedem Cluster in einem von der Farbe anhängigen Bereich liegt. Wir beweisen, dass unsere Algorithmen und Härteergebnisse sich im Wesentlichen auch auf diese weiter gefasste Fairnessdefinition übertragen lassen. Weiterhin beschreiben wir ein PTAS für Fair Correlation Clustering auf Wäldern, der im Gegensatz zur APX-Schwere von Correlation Clustering auf allgemeinen Graphen steht. Auch zeigen wir, dass wichtige Erkenntnisse über Fair Correlation Clustering in Wäldern auch für Biparte Graphen gelten, und hoffen damit, den Weg für zukünftige Algorithmen in allgemeineren Graphenklassen zu ebnen.

Acknowledgments

I like to thank my mentors, Katrin Casel and Martin Schirneck, for their constant support, helpful feedback, and great help during my work on this thesis. Further, I am very thankful towards all my family and friends for patiently listening to my monologues about clusterings and graphs. I like to especially thank Luise, who not only listened to more of these monologues than anybody else, but without her emotional support the work on this thesis would have been much more debilitating. Special thanks go to Leon and Wilhelm for their helpful proofreading and discussions.

Contents

1 Introduction

A reoccurring task when presented with a set of objects is to partition it into groups of *similar* or *connected* objects. Such a partition is called a *clustering*. Due to the many real-world use cases, countless variants of clusterings have been employed and analyzed. They differ in their way of defining the similarity of objects and the definition of what the properties of optimal clusterings are. More recently, fair clustering, a new line of research, has emerged that asks to find the optimal clustering only among the partitions that fulfill some fairness requirement. This is motivated by the various applications where the objects to be clustered have sensitive attributes that should not be allowed to be over- or underrepresented in any cluster. The task is then to find the optimum clustering such that each cluster has a certain distribution over the manifestations of the sensitive attributes.

In this thesis, we employ a common fairness definition that assigns a color to each object, representing some sensitive attribute. Then, a clustering is called fair if for each cluster and each color the ratio of objects of that color in the cluster approximately or exactly corresponds to the total ratio of vertices of that color. For example, imagine an airport security wants to find clusters among the travelers to assign each group another level of potential risk with corresponding anticipating measures. There are attributes like skin color that should not influence the assignment to a risk level. A bias in the data, however, may lead to some colors being over- or underrepresented in some clusters. Simply removing the skin color attribute from the data may not suffice as it may correlate with other attributes. Such problems are especially likely if one of the skin colors is far less represented in the data than others. A fair clustering finds the optimum clustering such that for each risk level the distribution of skin colors is fair, for example by requiring the distribution of each cluster to roughly match the distribution of skin colors among all travelers.

We study the fair variant of the well-known Correlation Clustering objective. There, we are given a graph in which two vertices are connected if and only if they are considered similar. The task is to find a partition of the set of

vertices that minimizes the number of disagreements, i.e., the number of pairs of vertices that are either placed in the same cluster but do not share an edge or are not placed in the same cluster though they share an edge.

For Fair Correlation Clustering, there are first results on approximating the best clustering. This is justified as Fair Correlation Clustering, just like the normal, unfair variant, is NP-hard. In this thesis, we investigate at which point Fair Correlation Clustering turns NP-hard. Correlation Clustering without the fairness constraint is easily solved on forests. Due to the sparseness of the graph, it suffices to build clusters of one or two vertices while cutting as few edges as possible, for example by employing a standard dynamic program. Such a strategy obtains the minimum cost as, assuming we place more than two vertices in a cluster in order to cut fewer edges, the benefit is compensated as then more pairs of vertices in the cluster are not connected because of the forest structure.

This raises the question of whether Fair Correlation Clustering is also solvable on forests. This thesis aims at providing answers to this question. We find that Fair Correlation Clustering is NP-hard even on restricted instances of forests and trees. However, we do not despair and further investigate whether under some additional assumptions there is still hope to give polynomial time algorithms. We aim at gaining a deeper understanding of the implications of the fairness constraint and what makes Fair Correlation Clustering so hard. Further, by giving polynomial time algorithms for specific instances, we hope to provide tools to tackle Fair Correlation Clustering and other fair clustering objectives in future work.

1.1 Outline and Contribution

We provide further insight into the hardness of Fair Correlation Clustering and algorithms for certain instances by analyzing the problem on forests and trees. First, we discuss the related work on fairness, Correlation Clustering and Fair Correlation Clustering in Section 1.2. In Chapter 2, we introduce the mathematical notation we require and provide precise definitions of the problem variants we examine. Then, in Chapter 3, we capture some helpful properties of minimum-cost fair clusterings on forests and the more general class of bipartite graphs. Using these, we give hardness proofs in Chapter 4. We find that Fair Correlation Clustering on trees is NP-hard, even when assuming a constant

diameter or constant maximum vertex degree. In Chapter 5, we contrast these hardness results with algorithms that solve Fair Correlation Clustering on forests and have polynomial running time under certain assumptions on the forests. In particular, we give complex dynamic programs that obtain a polynomial running time if either the expected size of the clusters or the number of clusters is constant. The size of the clusters in minimum-cost fair clusterings on forests is determined by the color ratio of the vertex set. To show that the hardness of the problem does not depend on the strict formulation of the fairness requirement, in Chapter 6, we transfer both hardness proofs and algorithms from Chapters 4 and 5 to a setting with a relaxed fairness constraint. In Chapter 7, we discuss approximation approaches to Fair Correlation Clustering on forests. Lastly, we evaluate our findings and propose directions for future work in Chapter 8. Tables 1.1 to 1.3 summarize our results.

Considering two colors in a ratio of $1:1$, we additionally prove that there is an algorithm solving α-relaxed Fair Correlation Clustering on forests in time in $O(n^{f(\alpha)})$ for some function f depending only on α (Theorem 6.12). Also, Theorem 5.1 does not only hold for forests but also gives that Fair Correlation Clustering is efficiently solvable in bipartite graphs with a color ratio of $1:1$.

Regarding approximation, we give a PTAS to Fair Correlation Clustering on forests (Theorem 7.3).

Diameter	Color ratio	Fairness	Trees	General Graphs
$\{2, 3\}$	$1:1$	exact	$O(n) \downarrow$	NP-hard Theorem 4.5
$\{2, 3\}$	any	exact	$O(n)$ Theorem 5.2	NP-hard $\uparrow$
4	$1:c$	exact/ relaxed	NP-hard Theorems 4.2 and 6.9	$\leftarrow$ NP-hard

Table 1.1: Hardness and running times of algorithms for (Relaxed) Fair Correlation Clustering by the diameter of the graph. An arrow indicates that a result is implied by a particular subset of instances.

Degree	Color ratio	Fairness	Trees	Forests
2	$1:c$	exact/ relaxed	?	NP-hard Theorems 4.1 and 6.8
2	$n/2$ colors, each 2 vertices	exact/ relaxed	NP-hard Theorem 4.4 and Corollary 6.5	$\leftarrow$ NP-hard
5	$1:c$	exact	NP-hard Theorem 4.3	$\leftarrow$ NP-hard

Table 1.2: Hardness and running times of algorithms for (Relaxed) Fair Correlation Clustering by the maximum degree of the graph. Results from forests directly carry over to general graphs. An arrow indicates that a result is implied by a particular subset of instances.

Color ratio	Forests	General Graphs
$1:1$	$n^{O(1)}$ Theorem 5.1	NP-hard [Ahm+20a]
$1:2$	$O(n^6)$ Theorem 5.5	?
$c_1 : c_2 : \ldots : c_k$	$O\left(n^{f(\sum_{i\in[k]} c_i)}\right)$ Theorem 5.6	NP-hard [Ahm+20a]
$1 : n/p - 1$	$O\left(n^{f'(p)}\right)$ Theorem 5.10	NP-hard $\downarrow$
$1 : \frac{n}{2} - 1$	$n^{O(1)}\uparrow$	NP-hard Theorem 4.6

Table 1.3: Hardness and running times of algorithms for Fair Correlation Clustering on forests and general graphs. f and f' are functions depending only on the given parameters and are specified in the respective theorems. Results from forests carry over to trees. An arrow indicates that a result is implied by a particular subset of instances.

1.2 Related Work

Here, we discuss the current state of research on Correlation Clustering, fair clustering, and Fair Correlation Clustering.

1.2.1 Clustering and Correlation Clustering

Clustering objectives and algorithms have been studied extensively. We refer to Xu and Tian [XT15] for a survey on clustering in general and to Schaeffer [Sch07] and Fortunato [For10] for surveys on graph clustering in particular.

The study of clustering objectives similar or identical to Correlation Clustering dates back to the 1960s [BSY99; Rég83; Zah64]. Bansal, Blum, and Chawla [BBC04] were the first to coin the term Correlation Clustering as a clustering objective. They prove NP-hardness and also propose a weighted form, where, for each pair of vertices, there is an edge with weight in $[-1, 1]$ stating how similar the vertices are. The most general formulation of Correlation Clustering regarding weights considers two positive real values for each pair of vertices, the first to be added to the cost if the objects are placed in the same cluster and the second to be added if the objects are placed in separate clusters [ACN08]. In the following, we focus on the unweighted variant. The state-of-the-art research on Correlation Clustering is summarized by the recent book by Bonchi, García-Soriano, and Gullo [BGG22], who also give a detailed introduction of the different variants regarding the weights and the cost function. Correlation Clustering has many applications for example in document clustering, biology, and genetics as well as in computer vision [BGG22].

We note that Correlation Clustering is an alternative formulation of Cluster Editing. There, the task is to transform the input graph into a cluster graph by adding and removing as few edges as possible. A cluster graph is a graph where each connected component is a clique. Hence, the obtained cluster graph defines a partition of the vertices and every added or removed edge corresponds to one pair of vertices incurring 1 unit to the Correlation Clustering cost.

In this thesis, we aim at finding the minimum-cost fair clustering on certain instances. With approximation algorithms, we also briefly discuss another approach. In general, Correlation Clustering is APX-hard [CGW05], meaning there is no algorithm giving an arbitrarily good approximation unless P = NP.

However, such an algorithm, a PTAS, exists for the maximum agree variant of Correlation Clustering [BBC04]. There, the task is to maximize the number of pairs of vertices that comply with the partition instead of minimizing the number of disagreements. While both objectives arrive at the same optimum clusterings, their approximations differ due to the different ways of computing the cost, either by counting disagreements or agreements. The best known approximation factor to (min-disagree) Correlation Clustering is 2.06 and is achieved by Chawla, Makarychev, Schramm, and Yaroslavtsev [Cha+15] by the use of new rounding schemes on a linear programming formulation for Correlation Clustering.

There is various research on solving Correlation Clustering efficiently and exactly under certain assumptions on the input graph. Komusiewicz and Uhlmann [KU12] show that Correlation Clustering remains NP-hard on graphs with maximum degree 6 or when each vertex may only be incident to up to 4 disagreements. Further, Bastos, Ochi, Protti, Subramanian, Martins, and Pinheiro [Bas+16] prove NP-hardness on graphs with diameter 2. Mannaa [Man10] states that Correlation Clustering is efficiently solved on unit interval graphs. Veldt, Wirth, and Gleich [VWG20] analyze Weighted Correlation Clustering in bipartite-like graphs. They find that if the vertex set partitions into two sets V_1, V_2 such that in each set all edges have the same weight and all edges between the sets have less weight, then solving Weighted Correlation Clustering is equivalent to computing a bipartite matching. They give approximations for other assumptions on the edge weights. For the unweighted case that considers only agreements and disagreements between the two vertex sets of a bipartite graph, Amit [Ami04] showed NP-hardness.

Some papers discuss incomplete Correlation Clustering and assume every edge in the input graph to be labeled by + or −, meaning it incurs a cost if the incident vertices are placed in separate or the same cluster, respectively. Pairs of vertices without an edge never incur any cost. Note that this differs from the way we define (complete) Fair Correlation Clustering in this thesis, where a missing edge corresponds to an edge labeled by −. For details on the definition of the complete Fair Correlation Clustering variant, see Section 2.2. The following two results refer to the graph structure of those incomplete Correlation Clustering instances and not to the graph structure as we use it. Xin gives a treewidth dynamic program that solves Weighted Correlation Clustering given a nice tree decomposition of width k of an n-vertex graph in

time in $O\big((k+1)^{k+2}n\big)$ [Xin11]. Bachrach, Kohli, Kolmogorov, and Zadimoghaddam [Bac+13] show that Weighted Correlation Clustering remains NP-hard in planar graphs. However, in our notation for complete Correlation Clustering, Berger, Grigoriev, and Winokurow [BGW17] provide a PTAS for planar graphs, contrasting the APX-hardness of Correlation Clustering on general graphs.

Correlation Clustering parametrized by the solution cost is *fixed-parameter tractable (FPT)*, i.e., there are algorithms solving Correlation Clustering in polynomial time in the graph size but exponential time in the Correlation Clustering cost of an optimum solution. The fastest known FPT-algorithm for Correlation Clustering was proposed by Boecker, Briesemeister, Bui, and Truss [Böc+09]. Their search tree approach runs in time in $O\big(1.82^k + n^3\big)$, where k is the solution cost and solves both the unweighted and weighted variant. Chen and Meng [CM12] give a kernel that transforms every input graph with n vertices and m edges into a graph of size at most $2k$ in time in $O(mn)$ such that solving Cluster Editing on the resulting graph is equivalent to solving Cluster Editing on the input graph. Bannach, Stockhusen, and Tantau [BST15] analyze Cluster Editing in a parallel-FPT setting. Hanaka and Lampis [HL22] discuss the Additively-Separable Hedonic Games problem on incomplete graphs, which can be regarded as a game-theoretic version of Weighted Correlation Clustering. Next to other lower bounds, they show NP-hardness even on stars and provide an algorithm parametrized by treewidth and diameter. Recall that the notion of stars, treewidth, and diameter here does not directly translate to our setting due to the different notation for incomplete graphs.

Regarding algorithms developed for practical purposes with no guarantee of a polynomial running time, Berg and Jaervisalo [BJ13] propose a solver based on MaxSAT. It is also able to solve incomplete Correlation Clustering. More solvers have been proposed for the 2021 PACE challenge on Cluster Editing [Kel+21].

Further, Bonchi, Gionis, Gullo, Tsourakakis, and Ukkonen [Bon+15] study Chromatic Correlation Clustering, a variant that colors the edges of the graph, other than Fair Correlation Clustering, which colors vertices. Each edge receives a label. The task is to find a clustering of the vertices and assign a label to each cluster. Then, the incurred cost of that clustering is the number of edges inside a cluster that do not match the label of the cluster plus the number of edges that are placed between different clusters (except they are labeled with the

special ℓ_0 label). It is a generalization of standard Correlation Clustering to which it is equivalent if every edge is either labeled by ℓ_0 or the only other label ℓ_1. Thereby, Chromatic Correlation Clustering is NP-hard. Klodt, Seifert, Zahn, Casel, Issac, and Friedrich [Klo+21] give a 3-approximation by analyzing the color-blind Pivot algorithm on Chromatic Correlation Clustering instances. Further, they give a color-sensitive heuristic and claim that it is better suited for many applications. Froese, Kellerhals, and Niedermeier [FKN22] consider Modification-Fair Cluster Editing. There, given a vertex-colored graph, the task is to solve Cluster Editing such that the number of edge modifications incident to vertices of each color has to be proportional to the number of vertices of that color. They prove the problem to be NP-hard even if there are only two colors and delete operations are not allowed. For the general case, when parametrized by the solution size, they give an FPT-algorithm that is linear in the graph size.

Recall that Correlation Clustering does not take a parameter for the number of clusters. While this is beneficial in many cases, in some others one wants to ensure that there is a fixed number of clusters. We find that in forests, all clusters in minimum-cost fair clusterings have a fixed size, which therefore is related to this variant of bounding the number of clusters. For every integer greater than 1, however, this bounded variant of Correlation Clustering is also NP-hard [SST04]. Other than for the APX-hard, unbounded variant, there is a known PTAS for this variant [GG06].

Due to the clusters of fixed size, Fair Correlation Clustering on forests is also related to k-Balanced Partitioning. There, the task is to partition the graph into k clusters of equal size while minimizing the number of edges that are cut by the partition. Feldman and Foschini [FF15] study this problem on trees and their results have interesting parallels with ours. They discuss an approach of first splitting the graph into clusters not exceeding a certain size and then smartly merging them into clusters of the desired size. We use a similar idea for our algorithms in Chapter 5. Further, just like us, they observe that hardness increases when switching from trees of diameter 3 to 4. Lastly, we reuse a construction proposed by Feldman and Foschini to show that not only k-Balanced Partitioning but also Fair Correlation Clustering is NP-hard on trees with maximum degree 5, see Theorem 4.3.

1.2.2 Fair Clustering

In the last decade, the notion of fairness in machine learning has increasingly attracted interest, see for example the review by Pessach and Schmueli [PS22]. Feldman, Friedler, Moeller, Scheidegger, and Venkatasubramanian [Fel+15] formalize fairness based on a US Supreme Court decision on disparate impact from 1971. It requires that sensitive attributes like gender or skin color should neither be explicitly considered in decision processes like hiring but also should the manifestations of sensitive attributes be proportionally distributed in all outcomes of the decision process. Feldman et al. formalized this notion for classification tasks. Chierichetti, Kumar, Lattanzi, and Vassilvitskii [Chi+17] adapted this concept for clustering tasks and formulated algorithms for the k-center and k-median objectives. Their work is seminal to the study of fair clustering algorithms. In this thesis, we employ the same disparate impact based understanding of fairness. Their algorithms are based on *fairlets*, minimal sets that satisfy the fairness constraint, i.e., have the same color (sensitive attribute) distribution as the whole data set. The principal idea is to first partition the data into fairlets and then apply a clustering approach to the contracted fairlets, as each combination of fair sets will still be fair. This work was extended by Bera, Chakrabarty, Flores, and Negahbani [Ber+19], who relax the fairness constraint in the sense of requiring upper and lower bounds on the representation of a color in each cluster and consider overlapping colors. Several works improve and extend their results in metric spaces [Bac+19; Ber+18; Esm+20; GSV22; KAM19; SSS20]. There are comparatively few results on fairness for graph clustering objectives. Ziko, Yuan, Granger, and Ayed [Zik+21] propose a different framework to capture fairness that works for both clustering objectives in metric spaces as well as in graphs. Vaichenker [Vai21] investigates Fair Ratio Cut. Dinitz, Srinivasan, Tsepenekas, and Vullikanti [Din+22] study Fair Disaster Containment, a graph cut problem involving fairness. Further, there are first results for Fair Correlation Clustering.

1.2.3 Fair Correlation Clustering

Fair Correlation Clustering has, to the best of our knowledge, attracted interest only quite recently. The results evolve around giving approximation algorithms for instances with limited numbers of colors.

Ahmadian, Epasto, Kumar, and Mahdian [Ahm+20b] initiated the research

on Fair Correlation Clustering. They employ a similar approach to the one of Chierichetti et al. [Chi+17] and first partition the graph into fairlets. Then, they contract each fairlet into a single vertex and add weighted edges between different fairlet-vertices depending on the number of edges between the two clusters. On the resulting graph, they approximate an (unfair) Correlation Clustering. Merging two fair clusters produces another fair cluster as the color ratio stays the same. Hence, they obtain a fair clustering. The quality of the solution depends on the quality of the fairlet decomposition as well as the approximation to unfair Correlation Clustering. Intuitively, they define the cost of a fairlet composition as the number of possible disagreements that are no longer represented in the contracted graph. Then, an β-approximation to unfair Correlation Clustering and an η-approximation to the fairlet decomposition gives an $(\beta(1+\eta)+\eta)$-approximation to Fair Correlation Clustering. The advantage of their approach is that using different fairlet decompositions allows for solving Fair Correlation Clustering for different fairness definitions. They analyze settings where the fairness constraint is given by some α and require that the ratio of each color in each cluster is at most α. They analyze 3 possibilities for α. For $\alpha = \frac{1}{2}$, which corresponds to our fairness understanding if there are two colors in a ratio of $1:1$, they obtain a 256-approximation. For $\alpha = \frac{1}{k}$, where k is the number of colors in the graph, they give a $16.48k^2$-approximation and for $\alpha = \frac{1}{t}$ for an arbitrary positive integer t, they show how to obtain a $O(\gamma t)$-approximation provided a γ-approximation to the $\frac{1}{t}$ fairlet composition. While it is interesting to see that the fairlet decomposition approach is applicable to Correlation Clustering as well, we note that the obtained approximations are quite far away from optimal solutions. In experiments on real-world data sets, they show that in both cases $\alpha = \frac{1}{2}$ and $\alpha = \frac{1}{k}$, the Correlation Clustering cost of the fair variant is slightly worse than the optimum unfair clustering while the tested unfair Correlation Clustering algorithms produce outputs with strong violations of the fairness requirement. We note that all their variants are only equivalent to our fairness notion if there are α^{-1} colors that all occur equally often.

The paper by Ahmadi, Galhotra, Saha, and Schwartz [Ahm+20a] is not to be confused with the one by Ahmadian et al. [Ahm+20b], though the papers are both from 2020 and share the same title *Fair Correlation Clustering*.

Ahmadi et al. provide an NP-hardness proof by a reduction from normal, unfair Correlation Clustering and give an $O(c^2)$-approximation algorithm

for instances with two colors in a ratio of $1 : c$. In the special case of a color ratio of $1 : 1$, they show how to approximate Fair Correlation Clustering using any approximation algorithm for unfair Correlation Clustering. This way, they obtain a $3\beta + 4$-approximation for Fair Correlation Clustering given any β-approximation to unfair Correlation Clustering. Using the state-of-the-art 2.06-approximation for unfair Correlation Clustering this gives a 10.18-approximation, clearly outperforming the 256-approximation by Ahmadian et al. They generalize their results such that for an instance with k colors in a ratio of $1 : c_2 : c_3 : \ldots : c_k$ for positive integers c_i, they give an $O(k^2 \cdot \max_{2 \leq i \leq k} c_i)$-approximation. Further, they study a relaxed fairness setting where for each color $1 \leq i \leq k$ there are positive integers q_i, p_i such that in the graph the ratio of color 1 to color i is between $1 : q_i$ and $1 : p_i$. Then, the task is to find the minimum-cost clustering such that in each cluster the ratio of each color i to color 1 is between $1 : q_i$ and $1 : p_i$. Ahmadi et al. give an $O(k^2 \cdot \max_{2 \leq i \leq k} q_i)$-approximation[1]. The underlying idea of their algorithms is to construct a bipartite graph by placing the vertices of one color on one side each, deleting edges between same colored vertices, and modifying the edge weights between vertices of different colors. Then, they solve normal, unfair Correlation Clustering on the right-hand side of the bipartite graph. Last, they compute a matching from the left side to the right side and assign vertices from the left to the clusters on the right according to the matching. If the ratio between the colors is not $1 : 1$ but $1 : c$, they compute a c-matching instead. They apply their algorithms to several real-world data sets and find that the clustering cost of Fair Correlation Clustering is only slightly worse while unfair Correlation Clustering produces clusters with very unbalanced color distributions.

Friggstad and Mousavi [FM21] provide the best known approximation to the $1 : 1$ color ratio case with a factor of 6.18 using linear programming relaxations. Further, they study Fair Correlation Clustering under local guarantees, i.e., finding the fair clustering that minimizes the maximum number of incident disagreement edges at any vertex.

To the best of our knowledge, the fourth and most recent publication on Fair Correlation Clustering is by Schwartz and Zats [SZ22]. They discuss

1 Their theorem states they achieve an $O(\max_{2 \leq i \leq k} q_i)$-approximation but when looking at the proof it seems they have accidentally forgotten the k^2 factor.

the more general case of incomplete Fair Correlation Clustering and give approximations for the max-agree variant.

2 Preliminaries

In this chapter, we define the mathematical notation we employ and give a formal definition to Fair Correlation Clustering.

2.1 Notation

We refer to the set of natural numbers $\{0, 1, 2, \ldots\}$ by $\mathbb{N}$. For $k \in \mathbb{N}$, let $[k] = \{1, 2, \ldots, k\}$ and $\mathbb{N}_{>k} = \mathbb{N} \setminus (\{0\} \cup [k])$. We write $2^{[k]}$ for the power set of $[k]$. By $\gcd(a_1, a_2, \ldots, a_k)$ we denote the *greatest common divisor* of $a_1, a_2 \ldots, a_k \in \mathbb{N}$.

An *undirected graph* $G = (V, E)$ is defined by a set of vertices V and a set of edges $E \subseteq \binom{V}{2} = \{\{u, v\} \mid u, v \in V, u \neq v\}$. If not stated otherwise, by the *size of* G we refer to $n + m$, where $n = |V|$ and $m = |E|$. A graph is called *complete* if $m = \frac{n(n-1)}{2}$. We call a graph $G = (A \cup B, E)$ *bipartite* if there are no edges in A nor B, i.e., $E \cap \binom{A}{2} = E \cap \binom{B}{2} = \emptyset$. For every $S \subseteq V$, we let $G[S] = \left(S, E \cap \binom{S}{2}\right)$ denote the *subgraph induced by* S. The *degree* of a vertex $v \in V$ is the number of edges incident to that vertex, $\delta(v) = |\{u \mid \{u, v\} \in E\}|$. The *degree* of a graph $G = (V, E)$ is the maximum degree of any of its vertices $\delta(G) = \max_{v \in V} \delta(v)$. A *path* of length k in G is a tuple of vertices $(v_1, v_2, \ldots, v_{k-1})$ such that for each $1 \leq i < k - 1$ we have $\{v_i, v_{i+1}\} \in E$. In this thesis, we only consider simple paths, i.e., we have $v_i \neq v_j$ for all $i \neq j$. A graph is called *connected* if for every pair of vertices u, v there is a path connecting u and v. The *distance* between two vertices is the length of the shortest path connecting these vertices and the *diameter* of a graph is the maximum distance between a pair of vertices. A *circle* is a path $(v_1, v_2, \ldots, v_k)$ such that $v_1 = v_k$ and $v_i \neq v_j$ only for all other pairs of $i \neq j$.

A *forest* is a graph without circles. A connected forest is called a *tree*. There is exactly one path connecting every pair of vertices in a tree. A tree is *rooted* by choosing any vertex $r \in V$ as the root. Then, every vertex v, except for the root, has a *parent*, which is the next vertex on the path from v to r. All vertices that have v as a parent are referred to as the *children* of v. A vertex without children is called a *leaf*. Given a rooted tree T, by T_v we denote the subtree induced by v

and its descendants, i.e., the set of vertices such that there is a path starting in v and ending in that vertex without using the edge to v's parent. Observe that each forest is a bipartite graph, for example by placing all vertices with even distance to the root of their respective tree on one side and the other vertices on the other side.

A finite set U can be *colored* by a function $c : U \to [k]$, for some $k \in \mathbb{N}_{>0}$. If there are only two colors, i.e., $k = 2$, for convenience we call them *red* and *blue*, instead by numbers.

For a *partition* $\mathcal{P} = \{S_1, S_2, \ldots, S_k\}$ with $S_i \cap S_j = \emptyset$ for $i \neq j$ of some set $U = S_1 \cup S_2 \cup \ldots \cup S_k$ and some $u \in U$ we use $\mathcal{P}[u]$ to refer to the set S_i for which $u \in S_i$. Further, we define the term *coloring* on sets and partitions. The *coloring of a set* counts the number of occurrences of each color in the set.

▶ **Definition 2.1 (Coloring of Sets).** Let S be a set colored by a function $c\colon S \to [k]$. Then, the coloring of S is an array C_S such that $C_S[i] = |\{s \in S \mid c(s) = i\}|$ for all $i \in [k]$. ◀

The *coloring of a partition* counts the number of occurrences of set colorings in the partition.

▶ **Definition 2.2 (Coloring of Partitions).** Let U be a colored set and let $\mathcal{P}$ be a partition of U. Let $\mathcal{C} = \{C_S \mid S \subseteq U\}$ denote the set of set colorings for which there is a subset of U with that coloring. By an arbitrarily fixed order, let $C_1, C_2, \ldots, C_\ell$ denote the elements of $\mathcal{C}$. Then, the coloring of $\mathcal{P}$ is an array $C_\mathcal{P}$ such that $C_\mathcal{P}[i] = |\{S \in \mathcal{P} \mid C_S = C_i\}|$ for all $i \in [\ell]$. ◀

2.2 Problem Definitions

In order to define Fair Correlation Clustering, we first give a formal definition of the unfair clustering objective. Correlation Clustering receives a pairwise similarity measure for a set of objects and aims at minimizing the number of similar objects placed in separate clusters and the number of dissimilar objects placed in the same cluster. For the sake of consistency, we reformulate the definition of Bonchi et al. [BGG22] such that the pairwise similarity between objects is given by a graph rather than an explicit binary similarity function. Given a graph $G = (V, E)$ and a partition $\mathcal{P}$ of V, the Correlation Clustering

cost is

$$\begin{aligned}\text{cost}(G, \mathcal{P}) = |\, \{\{u,v\} \in \tbinom{V}{2} \setminus E \mid \mathcal{P}[u] = \mathcal{P}[v]\}\,| + \\ |\, \{\{u,v\} \in E \mid \mathcal{P}[u] \neq \mathcal{P}[v]\}\,| \,.\end{aligned}$$

We refer to the first summand as the *intra-cluster cost* ψ and the second summand as the *inter-cluster cost* χ. Where G is clear from context, we abbreviate to $\text{cost}(\mathcal{P})$. Sometimes, we consider the cost of $\mathcal{P}$ on an induced subgraph. To this end, we allow the same cost definition as above also if $\mathcal{P}$ partitions some set $V' \supseteq V$. We define (unfair) Correlation Clustering as follows.

Correlation Clustering

Input: Graph $G = (V, E)$.
Task: Find a partition $\mathcal{P}$ of V that minimizes $\text{cost}(\mathcal{P})$.

We emphasize that this is the complete, unweighted, min-disagree form of Correlation Clustering. It is complete as every pair of objects is either similar or dissimilar but none is indifferent regarding the clustering. It is unweighted as the (dis)similarity between two vertices is binary. A pair of similar objects that are placed in separate clusters as well as a pair of dissimilar objects in the same cluster is called a *disagreement*, hence the naming of the min-disagree form. An alternative formulation would be the max-agree form with the objective to maximize the number of pairs that do not form a disagreement. Note that both formulations induce the same ordering of clusterings though approximation factors may differ because of the different formulations of the cost function.

This thesis tackles the Fair Correlation Clustering problem, which we define roughly following [Ahm+20b]. The fairness aspect limits the solution space to *fair* partitions. A partition is fair if each of its sets has the same color distribution as the universe that is partitioned.

▶ **Definition 2.3 (Fair Subset).** Let U be a finite set of elements colored by a function $c : U \to [k]$ for some $k \in \mathbb{N}_{>0}$. Let $U_i = \{u \in U \mid c(u) = i\}$ be the set of elements of color i for all $i \in [k]$. Then, some $S \subseteq U$ is fair if and only if for all colors $i \in [k]$ we have $\frac{|S \cap U_i|}{|S|} = \frac{|U_i|}{|U|}$. ◀

▶ **Definition 2.4 (Fair Partition).** Let U be a finite set of elements colored by a function $c : U \to [k]$ for some $k \in \mathbb{N}_{>0}$. Then, a partition $S_1 \cup S_2 \cup \ldots \cup S_\ell = U$ is fair if and only if all sets $S_1, S_2, \ldots, S_\ell$ are fair. ◀

With this, we define complete, unweighted, min-disagree Fair Correlation Clustering as follows.

Fair Correlation Clustering

Input: Graph $G = (V, E)$, coloring $c\colon V \to [k]$.
Task: Find a fair partition $\mathcal{P}$ of V that minimizes $\text{cost}(\mathcal{P})$.

In this thesis, when speaking of (Fair) Correlation Clustering, we refer to the complete, unweighted, min-disagree form, unless specified otherwise.

3 Implications of the Graph Structure

Here, we prove that in bipartite graphs and in forests in particular there is always a minimum-cost fair clustering such that all clusters are of some fixed size. This property is very useful, as it helps for building reductions in hardness proofs as well as algorithmic approaches that enumerate possible clusterings. Further, by the following lemma, this also implies that minimizing the inter-cluster cost suffices to minimize the Correlation Clustering cost, which simplifies the development of algorithms solving Fair Correlation Clustering on such instances.

▶ **Lemma 3.1.** Let $\mathcal{P}$ be a partition of the vertices of an m-edge graph G. Let χ denote the inter-cluster cost incurred by $\mathcal{P}$ on G. If all sets in the partition are of size d, then $\text{cost}(\mathcal{P}) = \frac{(d-1)n}{2} - m + 2\chi$. ◀

Proof. Note that in each of the $\frac{n}{d}$ clusters there are $\frac{d(d-1)}{2}$ pairs of vertices, each incurring an intra-cost of 1 if not connected by an edge. Let the total intra-cost be ψ. As there is a total of m edges, we have

$$\begin{aligned}\text{cost}(\mathcal{P}) &= \chi + \psi \\ &= \chi + \frac{n}{d} \cdot \frac{d(d-1)}{2} - (m - \chi) \\ &= \frac{(d-1)n}{2} - m + 2\chi.\end{aligned}$$

■

In particular, if G is a tree, this yields $\text{cost}(\mathcal{P}) = \frac{(d-3)n}{2} + 2\chi + 1$ as there $m = n - 1$.

3.1 Forests

We find that in forests, because of the small number of edges, there is always a minimum-cost partition such that all sets in the partition are of the minimum

size required to fulfill the fairness requirement. For example, if there are two colors of ratio $1 : 2$, then there is a minimum-cost clustering such that every cluster consists of exactly 1 vertex of the first color and 2 vertices of the second color, while every clustering with larger clusters (for example 2 vertices of the first color and 4 vertices of the second one) incurs at least the same cost. For all cases except two colors with equally many occurrences, the cost is even strictly greater.

▶ **Lemma 3.2.** Let F be a forest with $k \geq 2$ colors in a ratio of $c_1 : c_2 : \ldots : c_k$ with $c_i \in \mathbb{N}_{>0}$ for all $i \in [k]$, $\gcd(c_1, c_2, \ldots, c_k) = 1$, and $\sum_{i=1}^{k} c_i \geq 3$. Then, all clusters of every minimum-cost fair clustering are of size $\sum_{i=1}^{k} c_i$. ◀

Proof. Let $d = \sum_{i=1}^{k} c_i$. For any clustering $\mathcal{P}$ of V to be fair, all clusters must be at least of size d. We show that if there is a cluster S in the clustering with $|S| > d$, then we decrease the cost by splitting S. First note that in order to fulfill the fairness constraint, we have $|S| = ad$ for some $a \in \mathbb{N}_{\geq 2}$. Consider a new clustering $\mathcal{P}'$ obtained by splitting S into S_1, S_2, where $S_1 \subset S$ is an arbitrary fair subset of S of size d and $S_2 = S \setminus S_1$. Note that the cost incurred by every edge and non-edge with at most one endpoint in S is the same in both clusterings. Let ψ be the intra-cluster cost of $\mathcal{P}$ on $F[S]$. Regarding the cost incurred by the edges and non-edges with both endpoints in S, we know

$$\begin{aligned} \operatorname{cost}(F[S], \mathcal{P}) &\geq \psi \\ &\geq \frac{ad(ad-1)}{2} - (ad-1) \\ &= \frac{a^2d^2 - 3ad + 2}{2} \end{aligned}$$

since the cluster is of size ad and as it is part of a forest it contains at most $ad - 1$ edges. In the worst case, the $\mathcal{P}'$ cuts all the $ad - 1$ edges. However, we profit from the smaller cluster sizes. We have

$$\begin{aligned} \operatorname{cost}(F[S], \mathcal{P}') &= \chi + \psi \\ &\leq ad - 1 + \frac{d(d-1)}{2} + \frac{(a-1)d \cdot ((a-1)d - 1)}{2} \\ &= \frac{2d^2 + a^2d^2 - 2ad^2 + ad - 2}{2}. \end{aligned}$$

Hence, $\mathcal{P}'$ is cheaper by

$$\begin{aligned}\text{cost}(F[S],\mathcal{P}) - \text{cost}(F[S],\mathcal{P}') &\geq \frac{2ad^2 - 2d^2 - 4ad + 4}{2} \\ &= ad(d-2) - d^2 + 2.\end{aligned}$$

This term is increasing in a. As $a \geq 2$, by plugging in $a = 2$, we hence obtain a lower bound of

$$\text{cost}(F[S],\mathcal{P}) - \text{cost}(F[S],\mathcal{P}') \geq d^2 - 4d + 2.$$

For $d \geq 2$, the bound is increasing in d and it is positive for $d > 3$. This means, if $d > 3$ no clustering with a cluster of size more than d has minimal cost implying that all optimum clusterings only consist of clusters of size d.

Last, we have to argue the case $d = 3$, i.e., we have a color ratio of $1 : 2$ or $1 : 1 : 1$. In this case $d^2 - 4d + 2$ evaluates to -1. However, we obtain a positive change if we do not split arbitrarily but keep at least one edge uncut. Note that this means that one edge less is cut and one more edge is present, which means that our upper bound on $\text{cost}(T[S],\mathcal{P}')$ decreases by 2, so $\mathcal{P}$ is now cheaper. Hence, assume there is an edge $\{u,v\}$ such that $c(u) \neq c(v)$. Then by splitting S into $\{u,v,w\}$ and $S \setminus \{u,v,w\}$ for some vertex $w \in S \setminus \{u,v\}$ that makes the component $\{u,v,w\}$ fair, we obtain a cheaper clustering. If there is no such edge $\{u,v\}$, then $T[S]$ is not connected. This implies there are at most $3a - 3$ edges if the color ratio is $1 : 1 : 1$ since no edge connects vertices of different colors and there are a vertices of each color, each being connected by at most $a-1$ edges due to the forest structure. By a similar argument, there are at most $3a-2$ edges if the color ratio is $1 : 2$. Hence, the lower bound on $\text{cost}(T[S],\mathcal{P})$ increases by 1. At the same time, even if $\mathcal{P}'$ cuts all edges it cuts at most $3a-2$ times, so it is at least 1 cheaper than anticipated. Hence, in this case $\text{cost}(T[S],\mathcal{P}') < \text{cost}(T[S],\mathcal{P})$ no matter how we cut. ■

Note that Lemma 3.2 makes no statement about the case of two colors in a ratio of $1 : 1$. The statement does not hold in this case as illustrated in Figure 3.1. However, we prove that a slightly weaker statement holds not only for forests but for every bipartite graph, see Lemma 3.3.

Figure 3.1: Example forest where a cluster of size 4 and two clusters of size 2 incur the same cost. With one cluster of size 4 (left), the inter-cluster cost is 0 and the intra-cluster cost is 4. With two clusters of size 2 (right), both the inter-cluster and intra-cluster cost are 2.

3.2 Bipartite Graphs

We are able to partially generalize our findings for trees to bipartite graphs. We show that there is still always a minimum-cost fair clustering with cluster sizes fixed by the color ratio. However, in bipartite graphs there may also be minimum-cost clusterings with larger clusters. We start with the case of two colors in a ratio of $1:1$ and then generalize to other ratios.

▶ **Lemma 3.3.** Let $G = (A \cup B, E)$ be a bipartite graph with two colors in a ratio of $1:1$. Then, there is a minimum-cost fair clustering in G that has no clusters with more than 2 vertices. Further, each minimum-cost fair clustering can be transformed into a minimum-cost fair clustering such that all clusters contain no more than 2 vertices in linear time. If G is a forest, then no cluster in a minimum-cost fair clustering is of size more than 4. ◀

Proof. Note that, due to the fairness constraint, each fair clustering consists only of evenly sized clusters. We prove both statements by showing that in each cluster of at least 4 vertices there are always two vertices such that by splitting them from the rest of the cluster the cost does not increase and fairness remains.

Let $\mathcal{P}$ be a clustering and $S \in \mathcal{P}$ be a cluster with $|S| \geq 4$. Let $S_A = S \cap A$ and $S_B = S \cap B$. Assume there is $a \in S_a$ and $b \in S_b$ such that a and b have not the same color. Then, the clustering $\mathcal{P}'$ obtained by splitting S into $\{a, b\}$ and $S \setminus \{a, b\}$ is fair. We now analyze for each pair of vertices $u, v, u \neq v$ how the incurred Correlation Clustering cost changes. The cost does not change for every pair of vertices of which at most one vertex of u and v is in S. Further, it does not change if either $\{u, v\} = \{a, b\}$ or $\{u, v\} \subseteq S \setminus \{a, b\}$. There are at most

$$|S_A| - 1 + |S_B| - 1 = |S| - 2$$

edges with one endpoint in $\{a, b\}$ and the other in $S \setminus \{a, b\}$. Each of them is cut in $\mathcal{P}'$ but not in $\mathcal{P}$, so they incur an extra cost of at most $|S| - 2$. However, due to the bipartite structure, there are $|S_A| - 1$ vertices in $S \setminus \{a, b\}$ that have no edge to a and $|S_B| - 1$ vertices in $S \setminus \{a, b\}$ that have no edge to b. These $|S| - 2$ vertices incur a total cost of $|S| - 2$ in $\mathcal{P}$ but no cost in $\mathcal{P}'$. This makes up for any cut edge in $\mathcal{P}$, so splitting the clustering never increases the cost.

If there is no $a \in S_a$ and $b \in S_b$ such that a and b have not the same color, then either $S_A = \emptyset$ or $S_B = \emptyset$. In both cases, there are no edges inside S, so splitting the clustering in an arbitrary fair way never increases the cost.

By iteratively splitting large clusters in any fair clustering, we hence eventually obtain a minimum-cost fair clustering such that all clusters consist of exactly two vertices.

Now, assume G is a forest and there would be a minimum-cost clustering $\mathcal{P}$ with some cluster $S \in \mathcal{P}$ such that $|S| > 2a$ for some $a \in \mathbb{N}_{>2}$. Consider a new clustering $\mathcal{P}'$ obtained by splitting S into $\{u, v\}$ and $S \setminus \{u, v\}$, where u and v are two arbitrary vertices of different color that have at most 1 edge towards another vertex in S. There are always two such vertices due to the forest structure and because there are $\frac{S}{2}$ vertices of each color. Then, $\mathcal{P}'$ is still a fair clustering. Note that the cost incurred by each edge and non-edge with at most one endpoint in S is the same in both clusterings. Let ψ denote the intra-cluster cost of $\mathcal{P}$ in $G[S]$. Regarding the edges and non-edges with both endpoints in S, we know that

$$\begin{aligned}\operatorname{cost}(G[S], \mathcal{P}) &\geq \psi \\ &\geq \frac{2a(2a-1)}{2} - (2a-1) \\ &= 2a^2 - 3a + 1\end{aligned}$$

as the cluster consists of $2a$ vertices and has at most $2a - 1$ edges due to the forest structure. In the worst case, $\mathcal{P}'$ cuts 2 edges. However, we profit from the smaller cluster sizes. We have

$$\begin{aligned}\operatorname{cost}(G[S], \mathcal{P}') &\leq 2 + \psi \\ &\leq 2 + 1 + \frac{2(a-1)(2(a-1)-1)}{2} - (2a-1-2) \\ &= 2a^2 - 5a + 6.\end{aligned}$$

Hence, $\mathcal{P}$ costs at least $2a - 5$ more than $\mathcal{P}'$, which is positive as $a > 2$. Thus, in every minimum-cost fair clustering all clusters are of size 4 or 2. ■

We employ an analogous strategy if there is a different color ratio than $1 : 1$ in the graph. However, then we have to split more than 2 vertices from a cluster. To ensure that the clustering cost does not increase, we have to argue that we can take these vertices in some balanced way from both sides of the bipartite graph.

▶ **Lemma 3.4.** Let $G = (A \cup B, E)$ be a bipartite graph with $k \geq 2$ colors in a ratio of $c_1 : c_2 : \ldots : c_k$ with $c_i \in \mathbb{N}_{>0}$ for all $i \in [k]$ and $\gcd(c_1, c_2, \ldots, c_k) = 1$. Then, there is a minimum-cost fair clustering such that all its clusters are of size $d = \sum_{i=1}^{k} c_i$. Further, each minimum-cost fair clustering with larger clusters can be transformed into a minimum-cost fair clustering such that all clusters contain no more than d vertices in linear time. ◀

Proof. Due to the fairness constraint, each fair clustering consists only of clusters that are of size ad, where $a \in \mathbb{N}_{>0}$. We prove the statements by showing that a cluster of size at least $2d$ can be split such that the cost does not increase and fairness remains.

Let $\mathcal{P}$ be a clustering and $S \in \mathcal{P}$ be a cluster with $|S| = ad$ for some $a \geq 2$. Let $S_A = S \cap A$ as well as $S_B = S \cap B$ and w.l.o.g. $|S_A| \geq |S_B|$. Our proof consists of three steps.

- First, we show that there is a fair $\tilde{S} \subseteq S$ such that $|\tilde{S}| = d$ and $|\tilde{S} \cap A| \geq |\tilde{S} \cap B|$.
- Then, we construct a fair set $\hat{S} \subseteq S$ by replacing vertices in $\tilde{S}$ with vertices in $S_B \setminus \tilde{S}$ such that still $|\hat{S}| = d$, $|\hat{S}_A| \geq |\hat{S}_B|$, with $\hat{S}_A = \hat{S} \cap A$ and $\hat{S}_B = \hat{S} \cap B$, and additionally $|\hat{S}_A| - |\hat{S}_B| \leq |S_A| - |S_B|$.
- Last, we prove that splitting S into $\hat{S}$ and $S \setminus \hat{S}$ does not increase the clustering cost.

We then observe that the resulting clustering is fair, so the lemma's statements hold because any fair clustering with a cluster of more than d vertices is transformed into a fair clustering with at most the same cost, and only clusters of size d by repeatedly splitting larger clusters.

For the first step, assume there would be no such $\tilde{S} \subseteq S$, i.e., that we only could take $s < \frac{d}{2}$ vertices from S_A without taking more than c_i vertices of each color $i \in [k]$. Let s_i be the number of vertices of color i among these s vertices for all $i \in [k]$. Then, if $s_i = 0$ there is no vertex of color i in S_A as we could take the respective vertex into $\tilde{S}$, otherwise. Analogously, if $s_i < c_i$, then there are no more then s_i vertices of color i in S_A. If we take $s_i = c_i$ vertices, then up to all of the $ac_i = as_i$ vertices of that color are possibly in S_A. Hence,

$$|S_A| \leq \sum_{i=1}^{k} as_i = as < \frac{ad}{2}.$$

This contradicts $S_A \geq S_B$ because $|A| + |B| = ad$. Thus, there is a fair set $\tilde{S}$ of size d such that $|\tilde{S} \cap S_A| \geq |\tilde{S} \cap S_B|$.

Now, for the second step, we transform $\tilde{S}$ into $\hat{S}$. Note that, if $|S_A \setminus \tilde{S}| \geq |S_B \setminus \tilde{S}|$ it suffices to set $\hat{S} = \tilde{S}$. Otherwise, we replace some vertices from $\tilde{S} \cap S_A$ by vertices of the respective color from $S_B \setminus \tilde{S}$. We have to show that after this we still take at least as many vertices from S_A as from S_B and $|S_A| - |\hat{S}_A| \geq |S_B| - |\hat{S}_B|$. Let

$$\delta = |S_B \setminus \tilde{S}| - |S_A \setminus \tilde{S}| > 0.$$

Recall that $|S_A| \geq |S_B|$, so $\delta \leq |\tilde{S} \cap A| - |\tilde{S} \cap B|$. Then, we build $\hat{S}$ from $\tilde{S}$ by replacing $\frac{\delta}{2} \leq \frac{d}{2}$ vertices from $\tilde{S} \cap S_A$ with vertices of the respective color from $S_B \setminus \tilde{S}$. If there are such $\frac{\delta}{2}$ vertices, we have $|S_A \setminus \hat{S}_A| = |S_B \setminus \hat{S}_B|$ and $|\hat{S}_A| \geq |\hat{S}_B|$. Consequently, $\hat{S}$ fulfills the requirements.

Assume there would be no such $\frac{\delta}{2}$ vertices but that we could only replace $s < \frac{\delta}{2}$ vertices. Let s_i be the number of vertices of color i among these vertices for all $i \in [k]$. By a similar argumentation as above and because there are only $(a-1)c_i$ vertices of each color i in $S \setminus \hat{S}$, we have

$$\begin{aligned} |S_B \setminus \hat{S}| &\leq \sum_{i=1}^{k} (a-1)s_i \\ &= (a-1)s \\ &< \frac{(a-1)d}{2}. \end{aligned}$$

This contradicts $|S_B \setminus \tilde{S}| > |S_A \setminus \tilde{S}|$ as $|(S_A \cup S_B) \setminus \tilde{S}| = (a-1)d$. Hence, there are always enough vertices to create $\hat{S}$.

For the last step, we show that splitting S into $\hat{S}$ and $S \setminus \hat{S}$ does not increase the cost by analyzing the change for each pair of vertices $\{u,v\} \in \binom{V}{2}$. If not $u \in S$ and $v \in S$, the pair is not affected. Further, it does not change if either $\{u,v\} \subseteq \hat{S}$ or $\{u,v\} \subseteq (S \setminus \hat{S})$. For the remaining pairs of vertices, there are at most

$$|\hat{S}_A| \cdot |S_B \setminus \hat{S}_B| + |\hat{S}_B| \cdot |S_A \setminus \hat{S}_A| = |\hat{S}_A| \cdot |S_B| + |\hat{S}_B| \cdot |S_A| - 2\Big(|\hat{S}_A| \cdot |\hat{S}_B|\Big)$$

edges that are cut when splitting S into $\hat{S}$ and $S \setminus \hat{S}$. At the same time, there are

$$|\hat{S}_A| \cdot |S_A \setminus \hat{S}_A| + |\hat{S}_B| \cdot |S_B \setminus \hat{S}_B| = |\hat{S}_A| \cdot |S_A| + |\hat{S}_B| \cdot |S_B| - |\hat{S}_A|^2 - |\hat{S}_B|^2$$

pairs of vertices that are not connected and placed in separate clusters in $\mathcal{P}'$ but not in $\mathcal{P}$. Hence, we have $\mathcal{P}$ is more expansive than $\mathcal{P}'$ by at least

$$\begin{aligned}\operatorname{cost}(\mathcal{P}) - \operatorname{cost}(\mathcal{P}') &\geq |\hat{S}_A| \cdot |S_A| + |\hat{S}_B| \cdot |S_B| - |\hat{S}_A| \cdot |S_B| - |\hat{S}_B| \cdot |S_A| \\ &\quad - \Big(|\hat{S}_A|^2 - 2\Big(|\hat{S}_A| \cdot |\hat{S}_B|\Big) + |\hat{S}_B|^2\Big) \\ &\geq \Big(|\hat{S}_A| - |\hat{S}_B|\Big) \cdot (|S_A| - |S_B|) - \Big(|\hat{S}_A| - |\hat{S}_B|\Big)^2.\end{aligned}$$

This is non-negative as $|\hat{S}_A| \geq |\hat{S}_B|$ and $|\hat{S}_A| - |\hat{S}_B| \leq |S_A| - |S_B|$. Hence, splitting a cluster like this never increases the cost. ■

Unlike in forests, however, the color ratio yields no bound on the maximum cluster size in minimum-cost fair clusterings on bipartite graphs but just states there is a minimum-cost fair clustering with bounded cluster size. Let $G = (R \cup B, \{\{r,b\} \mid r \in R \wedge b \in B\})$ be a complete bipartite graph with $|R| = |B|$ such that all vertices in R are red and all vertices in B are blue. Then, all fair clusterings in G have the same cost, including the one with a single cluster $S = R \cup B$. This holds because of a similar argument as employed in the last part of Lemma 3.3 since every edge that is cut by a clustering is compensated for with exactly one pair of non-adjacent vertices that is then no longer in the same cluster.

4 Hardness Results

This chapter provides NP-hardness proofs for Fair Correlation Clustering under various restrictions.

4.1 Forests and Trees

With the knowledge of the fixed sizes of clusters in a minimum-cost clustering, we are able to show that the problem is surprisingly hard, even when limited to certain instances of forests and trees.

To prove the hardness of Fair Correlation Clustering under various assumptions, we reduce from the strongly NP-complete 3-Partition problem [GJ79].

3-Partition

Input: $n = 3p$ with $p \in \mathbb{N}$, positive integers $a_1, a_2, \ldots, a_n$, and $B \in \mathbb{N}$ such that $\frac{B}{4} < a_i < \frac{B}{2}$ as well as $\sum_{i=1}^{n} a_i = pB$.

Task: Decide if there is a partition of the numbers a_i into triples such that the sum of each triple is B.

Our first reduction yields hardness for many forms of forests.

▶ **Theorem 4.1.** Fair Correlation Clustering on forests with two colors in a ratio of $1 : c$ is NP-hard. It remains NP-hard when arbitrarily restricting the shape of the trees in the forest as long as for every $a \in \mathbb{N}$ it is possible to form a tree with a vertices. ◀

Proof. We reduce from 3-Partition. For every a_i we construct an arbitrarily shaped tree of a_i red vertices. Further, we let there be p isolated blue vertices. Note that the ratio between blue and red vertices is $1 : B$. We now show that there is a fair clustering $\mathcal{P}$ such that

$$\text{cost}(\mathcal{P}) = p \cdot \frac{B(B+1)}{2} - p(B-3)$$

if and only if the given instance is a yes-instance for 3-Partition.

If we have a yes-instance of 3-Partition, then there is a partition of the set of trees into p clusters of size B. By assigning the blue vertices arbitrarily to one unique cluster each, we hence obtain a fair partition. As there are no edges between the clusters and each cluster consists of $B+1$ vertices and $B-3$ edges, this partition has a cost of $p \cdot \frac{B(B+1)}{2} - p(B-3)$.

For the other direction, assume there is a fair clustering of cost $\frac{B(B+1)}{2} - p(B-3)$. By Lemma 3.2, each of the clusters consists of exactly one blue and B red vertices. Each cluster requires $\frac{B(B+1)}{2}$ edges, but the graph has only $p(B-3)$ edges. The intra-cluster cost alone is hence at least $p \cdot \frac{B(B+1)}{2} - p(B-3p)$. This means that the inter-cluster cost is 0, i.e., the partition does not cut any edges inside the trees. Since all trees are of size greater than $\frac{B}{4}$ and less than $\frac{B}{2}$, this implies that each cluster consists of exactly one blue vertex and exactly three uncut trees with a total of B vertices. This way, such a clustering gives a solution to 3-Partition, so our instance is a yes-instance.

As the construction of the graph only takes polynomial time in the instance size, this implies our hardness result. ■

Note that the hardness holds in particular for forests of paths, i.e., for forests with maximum degree 2.

With the next theorem, we adjust the proof of Theorem 4.1 to show that the hardness remains if the graph is connected.

▶ **Theorem 4.2.** Fair Correlation Clustering on trees with diameter 4 and two colors in a ratio of $1 : c$ is NP-hard. ◀

Proof. We reduce from 3-Partition. For every a_i we construct a star of a_i red vertices. Further, we let there be a star of p blue vertices. We obtain a tree of diameter 4 by connecting the center v of the blue star to all the centers of the red stars. The construction is depicted in Figure 4.1. Note that the ratio between blue and red vertices is $1 : B$. We now show that there is a fair clustering $\mathcal{P}$ such that

$$\text{cost}(\mathcal{P}) \leq \frac{pB^2 - pB}{2} + 7p - 7$$

if and only if the given instance is a yes-instance for 3-Partition.

If we have a yes-instance of 3-Partition, then there is a partition of the set of stars into p clusters of size B, each consisting of three stars. By assigning

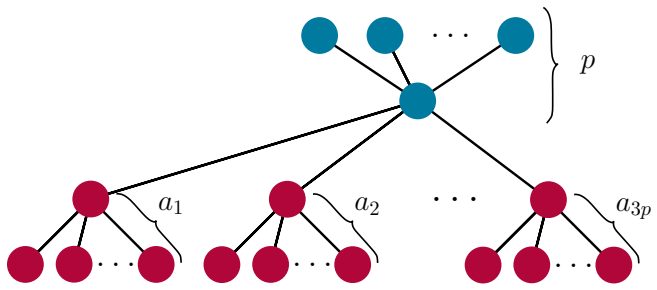

Figure 4.1: Constructed tree for the reduction from 3-Partition to Fair Correlation Clustering.

the blue vertices arbitrarily to one unique cluster each, we hence obtain a fair partition. We first compute the inter-cluster cost χ. We call an edge *blue* or *red* if it connects two blue or red vertices, respectively. We call an edge *blue-red* if it connects a blue and a red vertex. All $p-1$ blue edges are cut. Further, all edges between v (the center of the blue star) and red vertices are cut except for the three stars to which v is assigned. This causes $3p-3$ more cuts, so the inter-cluster cost is

$$\chi = 4p - 4.$$

Each cluster consists of $B+1$ vertices and $B-3$ edges, except for the one containing v which has B edges. The intra-cluster cost is hence

$$\begin{aligned}\psi &= p\left(\frac{B(B+1)}{2} - B + 3\right) - 3 \\ &= \frac{pB^2 - pB}{2} + 3p - 3.\end{aligned}$$

Combining the intra- and inter-cluster costs yields the desired cost of

$$\begin{aligned}\operatorname{cost}(\mathcal{P}) &= \chi + \psi \\ &= \frac{pB^2 - pB}{2} + 7p - 7.\end{aligned}$$

For the other direction, assume there is a fair clustering of cost at most $\frac{pB^2-pB}{2} + 7p - 7$. As there are $p(B+1)$ vertices, Lemma 3.2 gives that there are exactly p clusters, each consisting of exactly one blue and B red vertices. Let a denote the

number of red center vertices in the cluster of v. We show that $a = 3$. To this end, let χ_r denote the number of cut red edges. We additionally cut $p - 1$ blue and $3p - a$ blue-red edges. The inter-cluster cost of the clustering hence is

$$\chi = \chi_r + 4p - a - 1.$$

Regarding the intra-cluster cost, there are no missing blue edges and as v is the only blue vertex with blue-red edges, there are $(p-1)B + B - a = pB - a$ missing blue-red edges. Last, we require $p \cdot \frac{B(B-1)}{2}$ red edges, but the graph has only $pB - 3p$ red edges and χ_r of them are cut. Hence, there are at least $p \cdot \frac{B(B-1)}{2} - pB + 3p + \chi_r$ missing red edges, resulting in a total intra-cluster cost of

$$\psi \geq p \cdot \frac{B(B-1)}{2} + 3p + \chi_r - a.$$

This results in a total cost of

$$\begin{aligned} \text{cost}(\mathcal{P}) &= \chi + \psi \\ &\geq \frac{pB^2 - pB}{2} + 7p + 2\chi_r - 2a - 1. \end{aligned}$$

As we assumed $\text{cost}(\mathcal{P}) \leq \frac{pB^2 - pB}{2} + 7p - 7$, we have

$$2\chi_r - 2a + 6 \leq 0,$$

which implies $a \geq 3$ since $\chi_r \geq 0$. Additionally,

$$\chi_r \geq \frac{aB}{4} - (B - a),$$

because there are at least $\frac{B}{4}$ red vertices connected to each of the a chosen red centers but only a total of $B - a$ of them can be placed in their center's cluster. Thus, we have

$$\frac{aB}{2} - 2B + 6 = \frac{(a-4)B}{2} + 6 \leq 0,$$

implying $a < 4$ and proving our claim of $a = 3$. Further, as $a = 3$, we obtain $\chi_r \leq 0$, meaning that no red edges are cut, so each red star is completely contained in a cluster. Given that every red star is of size at least $\frac{B}{4}$ and at most $\frac{B}{2}$, this means each cluster consists of exactly three complete red stars with a total

number of B red vertices each and hence yields a solution to the 3-Partition instance.

As the construction of the graph only takes polynomial time in the instance size and the constructed tree is of diameter 4, this implies our hardness result. ■

The proofs of Theorems 4.1 and 4.2 follow the same idea as the hardness proof of [FF15, Theorem 2], which also reduces from 3-Partition to prove a hardness result on the k-Balanced Partitioning problem. There, the task is to partition the vertices of an uncolored graph into k clusters of equal size [FF15].

k-Balanced Partitioning

Input: Graph $G = (V, E)$, $k \in [n]$.

Task: Find a partition $\mathcal{P}$ of V that minimizes $|\{\{u,v\} \in E \mid \mathcal{P}[u] \neq \mathcal{P}[v]\}|$ under the constraint that $|\mathcal{P}| = k$ and $|S| \leq \lceil \frac{n}{k} \rceil$ for all $S \in \mathcal{P}$.

k-Balanced Partitioning is related to Fair Correlation Clustering on forests in the sense that the clustering has to partition the forest into clusters of equal sizes by Lemmas 3.2 and 3.3. Hence, on forests we can regard Fair Correlation Clustering as the fair variant of k-Balanced Partitioning. By [FF15, Theorem 8], k-Balanced Partitioning is NP-hard on trees of degree 5. In their proof, Feldmann and Foschini reduce from 3-Partition. We slightly adapt their construction to transfer the result to Fair Correlation Clustering.

▶ **Theorem 4.3.** Fair Correlation Clustering on trees of degree at most 5 with two colors in a ratio of $1 : c$ is NP-hard. ◀

Proof. We reduce from 3-Partition. 3-Partition remains strongly NP-hard when limited to instances where B is a multiple of 4 since for every instance we can create an equivalent instance by multiplying all integers by 4. Hence, assume a 3-Partition instance such that B is a multiple of 4. We construct a graph for Fair Correlation Clustering by representing each a_i for $i \in [n]$ by a gadget T_i. Each gadget has a center vertex that is connected to the end of five paths: one path of length a_i, three paths of length $\frac{B}{4}$, and one path of length $\frac{B}{4} - 1$. Then, for $i \in [n-1]$, we connect the dangling ends of the paths of length $\frac{B}{4} - 1$ in the gadgets T_i and T_{i+1} by an edge. So far, the construction is similar to the one by Feldmann and Foschini. We color all vertices added so far in red.

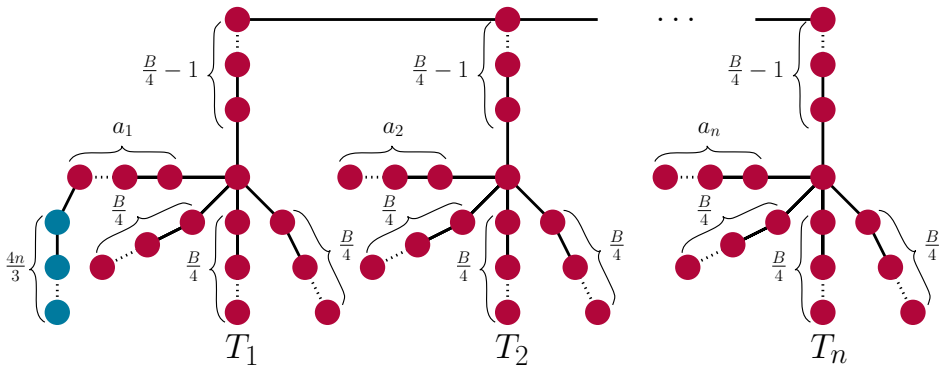

Figure 4.2: Constructed graph with degree 5 for the reduction from 3-Partition to Fair Correlation Clustering.

Then, we add a path of $\frac{4n}{3}$ blue vertices and connect it by an edge to an arbitrary vertex of degree 1. The resulting graph is depicted in Figure 4.2.

Note that the construction takes polynomial time and we obtain a graph of degree 5. We now prove that it has a fair clustering $\mathcal{P}$ such that

$$\text{cost}(\mathcal{P}) \leq \frac{(B-2)n}{2} + \frac{20n}{3} - 3$$

if and only if the given instance is a yes-instance for 3-Partition.

Assume we have a yes-instance for 3-Partition. We cut the edges connecting the different gadgets as well as the edges connecting the a_i-paths to the center of the stars. Then, we have n components of size B and 1 component of size a_i for each $i \in [n]$. The latter ones can be merged into $p = \frac{n}{3}$ clusters of size B without further cuts. Next, we cut all edges between the blue vertices and assign one blue vertex to each cluster. Thereby, note that the blue vertex that is already connected to a red cluster should be assigned to this cluster. This way, we obtain a fair clustering with inter-cluster cost

$$\begin{aligned}\chi &= n - 1 + n + \frac{4n}{3} - 1 \\ &= \frac{10n}{3} - 2,\end{aligned}$$

which, by Lemma 3.1, gives

$$\text{cost}(\mathcal{P}) = \frac{(B-2)n}{2} + \frac{20n}{3} - 3.$$

For the other direction, let there be a minimum-cost fair clustering $\mathcal{P}$ of cost at most $\frac{(B-2)n}{2} + \frac{20n}{3} - 3$. As $\sum_{i=1}^{n} a_i = \frac{nB}{3}$, the graph consists of $\frac{4n}{3} \cdot B$ red and $\frac{4n}{3}$ blue vertices. By Lemma 3.2, $\mathcal{P}$ hence consists of $\frac{4n}{3}$ clusters, each consisting of one blue vertex and B red vertices. Thus, $\mathcal{P}$ has to cut the $\frac{4n}{3} - 1$ edges on the blue path. Also, $\mathcal{P}$ has to partition the red vertices into sets of size B. By [FF15, Lemma 9] this requires at least $2n - 1$ cuts. This bounds the inter-cluster cost by

$$\chi \geq 2n - 1 + \frac{4n}{3} - 1 = \frac{10n}{3} - 2,$$

leading to a Correlation Clustering cost of $\frac{(B-2)n}{2} + \frac{20n}{3} - 3$ as seen above, so we know that no more edges are cut. Further, the unique minimum-sized set of edges that upon removal leaves no red components of size larger than B is the set of the $n - 1$ edges connecting the gadgets and the n edges connecting the a_i paths to the center vertices [FF15, Lemma 9]. Hence, $\mathcal{P}$ has to cut exactly these edges. As no other edges are cut, the a_i paths can be combined to clusters of size B without further cuts, so the given instance has to be a yes-instance for 3-Partition. ■

4.2 Fair Correlation Clustering on Paths

Theorem 4.1 yields that Fair Correlation Clustering is NP-hard even in a forest of paths. The problem when limited to instances of a single connected path is closely related to the Necklace Splitting problem [Alo87; AW86].

Discrete Necklace Splitting

Input: Opened necklace N, represented by a path of $n \cdot k$ beads, each in one of t colors such that for each color i there are $a_i \cdot k$ beads of that color for some $a_i \in \mathbb{N}$.

Task: Cut the necklace such that the resulting intervals can be partitioned into k collections, each containing the same number of beads of each color.

The only difference to Fair Correlation Clustering on paths, other than the naming, is that the number of clusters k is explicitly given. From Lemmas 3.2 and 3.3 we are implicitly given this value also for Fair Correlation Clustering, though. However, Alon and West do not constructively minimize the number of cuts required for a fair partition but non-constructively prove that there is always a partition of at most $(k-1)\cdot t$ cuts, if there are t colors and the partition is required to consist of exactly k sets with the same amount of vertices of each color. Thus, it does not directly help us when solving the optimization problem.

Moreover, Fair Correlation Clustering on paths is related to the 1-regular 2-colored variant of the Paint Shop Problem for Words (PPW). For PPW, a word is given as well as a set of colors, and for each symbol and color a requirement of how many such symbols should be colored accordingly. The task is to find a coloring that fulfills all requirements and minimizes the number of color changes between adjacent letters [EHO04].

Paint Shop Problem for Words (PPW)

Input: Word $w = w_1, w_2, \ldots, w_n \in \Sigma^*$, number of colors $k \in \mathbb{N}_{>0}$, and requirement function $r : \Sigma \times [k] \to \mathbb{N}$ such that for each symbol s used in w with $w[s]$ occurrences we have $\sum_{i=1}^{k} r(s,i) = w[s]$.

Task: Find an assignment function $f\colon [n] \to [k]$ of colors to the letters in w such that for each symbol $s \in \Sigma$ and color $i \in [k]$ the coloring fulfills the requirement function, i.e., $|\{j \in [n] \mid w_j = s \wedge f(j) = i\}| = r(s,i)$. The assignment f should minimze the number of color changes $|\{j \in [n-1] \mid f(j) \neq f(j+1)\}|$.

Let for example $w = aabab$ and $r(a,1) = 2, r(a,2) = r(b,1) = r(b,2) = 1$. Then, the assignment f with $f(1) = f(2) = f(3) = 1$ and $f(4) = f(5) = 2$ fulfills the requirement and has 1 color change.

PPW instances with a word containing every symbol exactly twice and two PPW-colors, each requiring one of each symbol, are called *1-regular 2-colored* and are shown to be NP-hard and even APX-hard [BEH06]. With this, we prove NP-hardness of Fair Correlation Clustering even on paths.

▶ **Theorem 4.4.** Fair Correlation Clustering on paths is NP-hard, even when limited to instances with exactly 2 vertices of each color. ◀

Proof. We reduce from 1-regular 2-colored PPW. Let $w = s_1 s_2, \ldots, s_\ell$. We represent the $\frac{\ell}{2}$ different symbols by $\frac{\ell}{2}$ colors and construct a path of length ℓ, where each type of symbol is represented by a unique color. By Lemma 3.2, any optimum Fair Correlation Clustering solution partitions the paths into two clusters, each containing every color exactly once, while minimizing the number of cuts (the inter-cluster cost) by Lemma 3.1. As this is exactly equivalent to assigning the letters in the word to one of two colors and minimizing the number of color changes, we obtain our hardness result. ■

APX-hardness however is not transferred since though there is a relationship between the number of cuts (the inter-cluster cost) and the Correlation Clustering cost, the two measures are not identical. In fact, as Fair Correlation Clustering has a PTAS on forests by Theorem 7.3, APX-hardness on paths would imply P = NP.

On a side note, observe that for every Fair Correlation Clustering instance on paths we can construct an equivalent PPW instance (though not all of them are 1-regular 2-colored) by representing symbols by colors and PPW-colors by clusters.

We note that it may be possible to efficiently solve Fair Correlation Clustering on paths if there are e.g. only two colors. There is an NP-hardness result on PPW with just two letters in [EHO04], but a reduction from these instances is not as easy as above since its requirements imply an unfair clustering.

4.3 Beyond Trees

By Theorem 4.2, Fair Correlation Clustering is NP-hard even on trees with diameter 4. Here, we show that if we allow the graph to contain circles, the problem is already NP-hard for diameter 2. Also, this nicely contrasts that Fair Correlation Clustering is solved on trees of diameter 2 in linear time, as we see in Section 5.1.

▶ **Theorem 4.5.** Fair Correlation Clustering on graphs of diameter 2 with two colors in a ratio of 1 : 1 is NP-hard. ◀

Proof. Cluster Editing, which is an alternative formulation of Correlation Clustering, is NP-hard on graphs of diameter 2 [Bas+16]. Further, Ahmadi et al. [Ahm+20a] give a reduction from Correlation Clustering to Fair

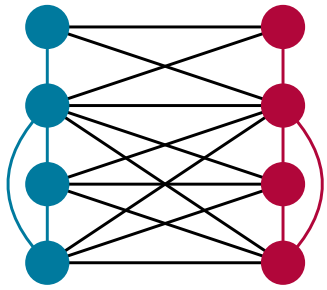

Figure 4.3: Graph as constructed by Ahmadi et al. [Ahm+20a] for the reduction from Correlation Clustering to Fair Correlation Clustering. The blue vertices and edges correspond to the original graph $G = (V, E)$, red vertices and edges to its mirror, i.e., V' and E', and black edges to $\tilde{E}$.

Correlation Clustering with a color ratio of $1 : 1$. They show that one can solve Correlation Clustering on a graph $G = (V, E)$ by solving Fair Correlation Clustering on the graph $G' = (V \cup V', E \cup E' \cup \tilde{E})$ that mirrors G. The vertices in V are colored blue and the vertices in V' are colored red. Formally, $V' = \{u' \mid u \in V\}$ and $E' = \{\{u', v'\} \mid \{u, v\} \in E\}$. Further, $\tilde{E}$ connects every vertex with its mirrored vertex as well as the mirrors of adjacent vertices, i.e., $\tilde{E} = \{\{u, u'\} \mid u \in V\} \cup \{\{u, v'\} \mid u \in V \wedge v' \in V' \wedge \{u, v\} \in E\}$, see Figure 4.3.

Observe that if G has diameter 2 then G' also has diameter 2 as follows. As every pair of vertices $\{u, v\} \in \binom{V}{2}$ is of maximum distance 2 and the vertices as well as the edges of G are mirrored, every pair of vertices $\{u', v'\} \in \binom{V'}{2}$ is of maximum distance 2. Further, every vertex and its mirrored vertex have a distance of 1. For every pair of vertices $u \in V, v' \in V'$ we distinguish two cases. If $\{u, v\} \in E$, then $\{u, v'\} \in \tilde{E}$, so the distance is 1. Otherwise, as the distance between u and v is at most 2 in G, there is $w \in V$ such that $\{u, w\} \in E$ and $\{v, w\} \in E$. Thus, $\{u, w'\} \in \tilde{E}$ and $\{w', v'\} \in E'$, so the distance of u and v' is at most 2.

As Correlation Clustering on graphs with diameter 2 is NP-hard and the reduction by Ahmadi et al. [Ahm+20a] constructs a graph of diameter 2 if the input graph is of diameter 2, we have proven the statement. ■

Further, we show that on general graphs Fair Correlation Clustering is NP-hard, even if the colors of the vertices allow for no more than 2 clusters in any fair

clustering. This contrasts our algorithm in Section 5.4 solving Fair Correlation Clustering on forests in polynomial time if the maximum number of clusters is constant. To this end, we reduce from the NP-hard Bisection problem [GJ79], which is the $k = 2$ case of k-Balanced Partitioning.

Bisection

Input: Graph $G = (V, E)$.

Task: Find a partition $\mathcal{P} = \{A, B\}$ of V that minimizes $|\{\{u, v\} \in E \mid u \in A \wedge v \in B\}|$ under the constraint that $|A| = |B|$.

▶ **Theorem 4.6.** Fair Correlation Clustering on graphs with two colors in a ratio of $1 : c$ is NP-hard, even if $c = \frac{n}{2} - 1$ and the graph is connected. ◀

Proof. We reduce from Bisection. Let $G = (V, E)$ be a Bisection instance and assume it has an even number of vertices (otherwise it is a trivial no-instance). The idea is to color all of the vertices in V red and add two cliques, each consisting of one blue and $|V|$ red vertices to enforce that a minimum-cost Fair Correlation Clustering consists of exactly two clusters and thereby partitions the vertices of the original graph in a minimum-cost bisection. The color ratio is $2 : 3|V|$ which equals $1 : \frac{|V'|}{2} - 1$ with V' being the set of the newly constructed graph. We have to rule out the possibility that a minimum-cost Fair Correlation Clustering is just one cluster containing the whole graph. We do this by connecting the new blue vertices v_1, v_2 to only one arbitrary red vertex $v \in V$. We illustrate the scheme in Figure 4.4. We first argue that every clustering with two clusters is cheaper than placing all vertices in the same cluster. Let $n = |V|$ as well as $m = |E|$. Let $\mathcal{P}$ be a clustering that places all vertices in a single cluster. Then,

$$\begin{aligned} \text{cost}(\mathcal{P}) &= \frac{(3n+2)(3n+1)}{2} - \left(m + 2 + 2 \cdot \frac{n(n+1)}{2}\right) \\ &= \frac{7n^2}{2} + \frac{7n}{2} - m - 1, \end{aligned}$$

as the cluster is of size $3n+2$, there is a total of $m+2$ plus the edges of the cliques, and no edge is cut. Now assume we have a clustering $\mathcal{P}'$ with an inter-cluster

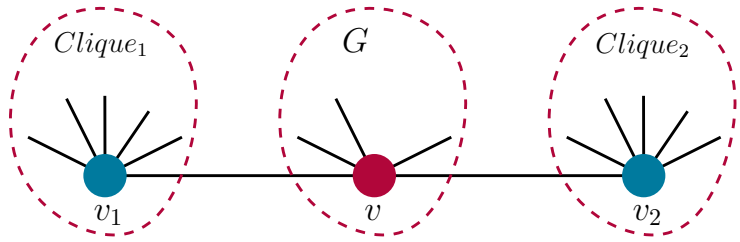

Figure 4.4: Graph constructed for the reduction from Bisection to a Fair Correlation Clustering instance with just 2 large clusters. The middle part corresponds to the input graph G and is colored red. $Clique_1$ and $Clique_2$ are both cliques of $|V|$ red vertices and one blue vertex each.

cost of χ' that puts each clique in a different cluster. Then,

$$\begin{aligned}\text{cost}(\mathcal{P}') &= \chi' + 2 \cdot \frac{(\frac{3n}{2}+1)(\frac{3n}{2})}{2} - \left(m - \chi' + \frac{n(n+1)}{2}\right) \\ &= \frac{7n^2}{4} + n - m + 2\chi' \\ &\leq \frac{9n^2}{4} + n - m + 2,\end{aligned}$$

since there are at most $\frac{n}{2} \cdot \frac{n}{2}$ inter-cluster edges between vertices of V and one inter-cluster edge from v to either v_1 or v_2, so $\chi \leq \frac{n^2}{4} + 1$. Placing all vertices in the same cluster is hence more expansive by

$$\begin{aligned}\text{cost}(\mathcal{P}) - \text{cost}(\mathcal{P}') &\geq \frac{7n^2}{2} + \frac{7n}{2} - m - 1 - \left(\frac{9n^2}{4} + n - m + 2\right) \\ &= \frac{5n^2}{4} + \frac{5n}{2} - 3\end{aligned}$$

than any clustering with two clusters. This is positive for $n \geq 2$. Thus, Fair Correlation Clustering will always return at least two clusters. Also, due to the fairness constraint and there being only two blue vertices, it creates exactly two clusters.

Further, it does not cut vertices from one of the two cliques for the following reason. As the clusters are of fixed size, by Lemma 3.1 we can focus on the inter-cluster cost to argue that a minimum-cost Fair Correlation Clustering

only cuts edges in E. First, note that it is never optimal to cut vertices from both cliques as just cutting the difference from one clique cuts fewer edges. This also implies that at most $\frac{n}{2}$ red vertices are cut from the clique as otherwise, the other cluster would have more than the required $\frac{3n}{2}$ red vertices. So, assume $0 < a \leq \frac{n}{2}$ red vertices are cut from one clique. Any such solution has an inter-cluster cost of $a \cdot (n + 1 - a) + \chi_E$, where χ_E is the number of edges in E that are cut to split V into two clusters of size $\frac{n}{2} + a$ and $\frac{n}{2} - a$ as required to make a fair partition. We note that by not cutting the cliques and instead cutting off a vertices from the cluster of size $\frac{n}{2} + a$, we obtain at most $a \cdot \frac{n}{2} + \chi_E$ cuts. As $\frac{n}{2} < n + 1 - a$, this implies that no optimal solution cuts the cliques. Hence, every optimum solution partitions the vertices in V in a minimum-cost bisection.

Thus, by solving Fair Correlation Clustering on the constructed graph we can solve Bisection in G. As further, the constructed graph is of polynomial size in $|V|$, we obtain our hardness result. ■

5 Algorithms for Forests

The results from Chapter 4 make it unlikely that there is a polynomial time algorithm solving Fair Correlation Clustering on trees as this would imply P = NP. However, we are able to give polynomial time algorithms for certain instances of forests.

5.1 Convenient Instances

First, we observe that Fair Correlation Clustering on forests and even on bipartite graphs is equivalent to the Maximum Bipartite Matching problem and thereby efficiently solvable if there are just two colors that occur equally often. This is due to there being a minimum-cost fair clustering such that each cluster consists of exactly one vertex of each color.

▶ **Theorem 5.1.** Computing a minimum-cost fair clustering on bipartite graphs with two colors in a ratio of 1:1 is as hard as finding a maximum bipartite matching. ◀

Proof. Let the colors be red and blue. By Lemma 3.3, there is an optimum clustering for which all clusters are of size at most 2. Due to the fairness constraint, each such cluster consists of exactly 1 red and 1 blue vertex. By Lemma 3.1, the lowest cost is achieved by the lowest inter-cluster cost, i.e., when the number of clusters where there is an edge between the two vertices is maximized. This is exactly the matching problem on the bipartite graph $G' = (R \cup B, E')$, with R and B being the red and blue vertices, respectively, and $E' = \{\{u, v\} \in E \mid u \in R \wedge v \in B\}$. After computing an optimum matching, each edge of the matching defines a cluster and unmatched vertices are packed into fair clusters arbitrarily.

For the other direction, if we are given an instance $G' = (R \cup B, E')$ for bipartite matching, we color all the vertices in R red and the vertices in B blue. Then, a minimum-cost fair clustering is a partition that maximizes the number of edges in each cluster as argued above. As each vertex is part of exactly one cluster and all clusters consist of one vertex in R and one vertex in B, this corresponds to a maximum bipartite matching in G'. ■

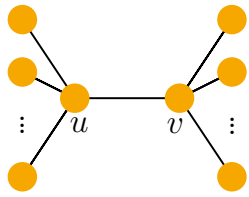

Figure 5.1: Shape of every tree with diameter 3.

As the Hopcroft-Karp-Karzanov algorithm solves maximum bipartite matching on n-vertex m-edge graphs in time in $O(m\sqrt{n})$ [HK71], Theorem 5.1 gives that Fair Correlation Clustering on forests with two colors in a ratio of $1:1$ is solved in the same asymptotic running time.

Next, recall that Theorem 4.2 states that Fair Correlation Clustering on trees with a diameter of at least 4 is NP-hard. With the next theorem, we show that we efficiently solve Fair Correlation Clustering on trees with a diameter of at most 3, so our threshold of 4 is tight unless P = NP.

▶ **Theorem 5.2.** Fair Correlation Clustering on trees with a diameter of at most 3 can be solved in linear time. ◀

Proof. Diameters of 0 or 1 are trivial and the case of two colors in a ratio of $1:1$ is handled by Theorem 5.1. So, assume $d > 2$ to be the minimum size of a fair cluster. A diameter of two implies that the tree is a star. In a star, the inter-cluster cost equals the number of vertices that are not placed in the same cluster as the center vertex. By Lemma 3.2, every clustering of minimum cost has minimum-sized clusters. As in a star, all these clusterings incur the same inter-cluster cost of $n - d + 1$ they all have the same Correlation Clustering cost by Lemma 3.1. Hence, outputting any fair clustering with minimum-sized clusters solves the problem. Such a clustering can be computed in time in $O(n)$.

If we have a tree of diameter 3, it consists of two adjacent vertices u, v such that every vertex $w \in V \setminus \{u, v\}$ is connected to either u or v and no other vertex, see Figure 5.1. This is due to every graph of diameter 3 having a path of four vertices. Let the two in the middle be u and v. The path has to be an induced path or the graph would not be a tree. We can attach other vertices to u and v without changing the diameter but as soon as we attach a vertex elsewhere, the diameter increases. Further, there are no edges between vertices in $V \setminus \{u, v\}$ as the graph would not be circle-free.

For the clustering, there are now two possibilities, which we try out separately. Either u and v are placed in the same cluster or not. In both cases, Lemma 3.2 gives that all clusters are of minimal size d. If u and v are in the same cluster, all clusterings of fair minimum sized clusters incur an inter-cluster cost of $n-d+2$ as all but $d-2$ vertices have to be cut from u and v. In $O(n)$, we greedily construct such a clustering $\mathcal{P}_1$. If we place u and v in separate clusters, the minimum inter-cluster is achieved by placing as many of their respective neighbors in their respective clusters as possible. After that, all remaining vertices are isolated and are used to make these two clusters fair and if required form more fair clusters. Such a clustering $\mathcal{P}_2$ is also computed in $O(n)$. We then return the cheaper clustering. This is a fair clustering of minimum cost as either u and v are placed in the same cluster or not, and for both cases, $\mathcal{P}_1$ and $\mathcal{P}_2$ are of minimum cost, respectively. ■

5.2 Color Ratio 1 : 2

We now aim at giving algorithms for Fair Correlation Clustering on forests that do not require a certain diameter or degree. As a first step to solve these less restricted instances, we develop an algorithm to solve Fair Correlation Clustering on forests with a color ratio of 1 : 2.

Let w.l.o.g. the vertices be colored blue and red with twice as many red vertices as blue ones. We call a connected component of size 1 a *b-component* or *r-component*, depending on whether the contained vertex is blue or red. Analogously, we apply the terms *br-component, rr-component*, and *brr-component* to components of size 2 and 3.

5.2.1 Linear Time Approach

Because of Lemma 3.2, we know that in every minimum-cost fair clustering each cluster contains exactly 1 blue and 2 red vertices. Our high-level idea is to employ two phases.

In the first phase, we partition the vertices of the forest F in a way such that in every cluster there are at most 1 blue and 2 red vertices. We call such a partition a *splitting* of F. We like to employ a standard tree dynamic program that bottom-up collects vertices to be in the same connected component and cuts edges if otherwise there would be more than 1 blue or 2 red vertices in the

component. We have to be smart about which edges to cut, but as only up to 3 vertices can be placed in the topmost component, we have only a limited number of possibilities we have to track to find the splitting that cuts the fewest edges.

After having found that splitting, we employ a second phase, which finds the best way to assemble a fair clustering from the splitting by merging components and cutting as few additional edges as possible. As, by Lemma 3.1, a fair partition with the smallest inter-cluster cost has a minimum Correlation Clustering cost, this would find a minimum-cost fair clustering.

Sadly, this approach does not work that easily. We find that the number of cuts incurred by the second phase depends on the number of br- and r-components.

▶ **Lemma 5.3.** Let $F = (V, E)$ be an n-vertex forest with colored vertices in blue and red in a ratio of $1 : 2$. Let in each connected component be at most 1 blue vertex and at most 2 red vertices. Let $\#(br)$ and $\#(r)$ be the number of br- and r-components, respectively. Then, after cutting $\max\{0, \frac{\#(br)-\#(r)}{2}\}$ edges, the remaining connected components can be merged such that all clusters consist of exactly 1 blue and 2 red vertices. Such a set of edges can be found in time in $O(n)$. Further, when cutting less than $\max\{0, \frac{\#(br)-\#(r)}{2}\}$ edges, such merging is not possible. ◀

Proof. As long as possible, we arbitrarily merge b-components with rr-components as well as br-components with r-components. For this, no edges have to be cut. Then, we split the remaining rr-components and merge the resulting r-components with one br-component each. This way, we incur $\max\{0, \frac{\#(br)-\#(r)}{2}\}$ more cuts and obtain a fair clustering as now each cluster contains two red and one blue vertex. This procedure is done in time in $O(n)$.

Further, there is no cheaper way. For each br-component to be merged without further cuts we require an r-component. There are $\#(r)$ r-components and each cut creates either at most two r-components or one r-component while removing a br-component. Hence, $\max\{0, \frac{\#(br)-\#(r)}{2}\}$ cuts are required. ■

For our approach to work, the first phase has to simultaneously minimize the number of cuts as well as the difference between br- and r-components. This is, however, not easily possible. Consider the tree in Figure 5.2.

There, with one additional cut edge we have three br-components less and one r-component more. Using a standard tree dynamic program, therefore, does not suffice as when encountering the tree as a subtree of some larger forest or tree, we would have to decide between optimizing for the number of cut edges or

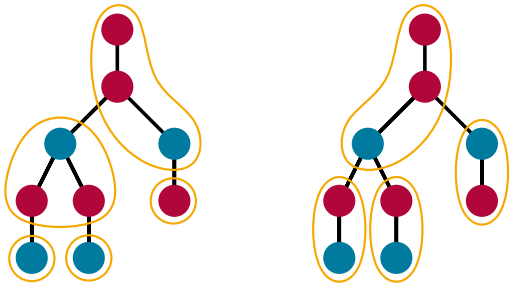

Figure 5.2: A tree for which the splitting with the minimum number of cuts (right) has 3 more br-components and 1 less r-component than a splitting with one more cut edge (left).

the difference between br- and r-components. There is no trivial answer here as the choice depends on how many br- and r-components are obtained in the rest of the graph. For our approach to work, we hence have to track both possibilities until we have seen the complete graph, setting us back from achieving a linear running time.

5.2.2 Join-subroutine

In the first phase, we might encounter situations that require us to track multiple ways of splitting various subtrees. When we reach a parent vertex of the roots of these subtrees, we join these various ways of splitting. For this, we give a subroutine called Join. We first formalize the output by the following lemma, then give an intuition on the variables, and lastly prove the lemma by giving the algorithm.

▶ **Lemma 5.4.** Let $R_1, R_2, \ldots, R_{\ell_1}$ for $\ell_1 \in \mathbb{N}_{>1}$ with $R_i \in (\mathbb{N} \cup \{\infty\})^{\ell_2}$ for $\ell_2 \in \mathbb{N}, i \in [\ell_1]$ and f be a computable function $f\colon [\ell_2] \times [\ell_2] \to 2^{[\ell_2]}$. For $x \in [\ell_2]$, let

$$A_x = \{M \in ([\ell_2])^{\ell_1} \mid x \in \hat{f}(M[1], M[2], \ldots, M[\ell_2])\},$$

whereby for all $x_1, x_2, \ldots \in [\ell_2]$

$$\hat{f}(x_1, x_2) = f(x_1, x_2)$$

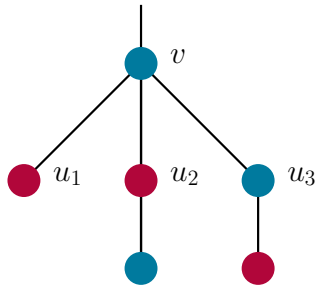

Figure 5.3: Exemplary graph for a Join-subroutine.

and for all $2 \leq k \leq \ell_2$

$$\hat{f}(x_1, x_2, \ldots, x_k) = \bigcup_{x \in \hat{f}(x_1, x_2, \ldots, x_{k-1})} f(x, x_k).$$

Then, an array $R \in (\mathbb{N} \cup \{\infty\})^{\ell_2}$ such that $R[x] = \min_{M \in A_x} \sum_{i=1}^{\ell_1} R_i[M[i]]$ for all $x \in [\ell_2]$ can be computed in time in $\mathrm{O}\big(\ell_1 \cdot \ell_2^2 \cdot T_f\big)$, where T_f is the time required to compute f. ◀

As we later reuse the routine, it is formulated more generally than required for this section. Here, for the $1:2$ case, assume we want to join the splittings of the children $u_1, u_2, \ldots, u_{\ell_1}$ of some vertex v. For example, assume v has three children as depicted in Figure 5.3. Then, for each child u_i, let there be an array R_i such that $R_i[x]$ is the minimum number of cuts required to obtain a splitting of the subtree T_{u_i} that has exactly x more br-components than r-components. For our example, assume all edges between v and its children have to be cut. We see, that $R_1[-1] = 1$ and $R_1[x] = \infty$ for $x \neq -1$, as the only possible splitting for the subtree of u_1 cuts only the edge to v and has one more r-component than br-components. Further, we have $R_2[1] = 1$ (by only cutting $\{v, u_2\}$), $R_2[-1] = 2$ (by cutting both edges of u_2), and $R_2[x] = \infty$ for $x \notin \{-1, 1\}$. Last, note that $R_3 = R_2$.

The function f returns the set of indices that should be updated when merging two possibilities. When a splitting of one child's subtree has x_1 more br-components and a splitting of another child's subtree has x_2 more br-components, then the combination of these splittings has $x_1 + x_2$ more br-components than r-components. Hence, the only index to update is $f(x_1, x_2) = \{x_1 + x_2\}$. Later,

we will require to update more than a single index, so f is defined to return a set instead of a single index. Note that by the definition of f and $\hat{f}$, each value placed in $R[x]$ by the routine corresponds to choosing exactly one splitting from each array R_i such that the total difference between br-components and r-components sums up to exactly x.

In our example, assume any splitting is chosen for each of the three subtrees. Let x_i denote the difference of br- and r-components of the chosen splitting for the subtree rooted at u_i for $1 \leq i \leq 3$. Then, Join sets $R[x]$ for $x = x_1 + x_2 + x_3$. If there are multiple ways to achieve an index x, the one with the minimum number of cuts is stored in $R[x]$. In the example, we have 4 possibilities, as $x_1 = -1$ and $x_2, x_3 \in \{-1, 1\}$. Note that $x_1 = -1, x_2 = -1, x_3 = 1$ and $x_1 = -1, x_2 = 1, x_3 = -1$ both evaluate to $x = -1$. Hence, only one of the two combinations is stored (the one with fewer cuts, here an arbitrary one as both variants imply 4 cuts). For the resulting array R, we have $R[-3] = 5, R[-1] = 4, R[1] = 3$, and $R[x] = \infty$ for $x \notin \{-3, -1, 1\}$. Observe that the numbers of cuts in R correspond to the sums of the numbers of cuts in the subtrees for the respective choice of x_i.

In the following, we give how the Join-subroutine is computed.

Proof of Lemma 5.4. The algorithm works in an iterative manner. Assume it has found the minimum value for all indices using the first $i - 1$ arrays and they are stored in R^{i-1}. It then *joins* the i-th array by trying every index x_1 in R^{i-1} with every index x_2 in R_i. Each time, for all indices $x \in f(x_1, x_2)$, it sets $R^i[x]$ to $R^{i-1}[x_1] + R_i[x_2]$ if it is smaller than the current element there. Thereby, it tries all possible ways of combining the interim solution with R_i and for each index tracks the minimum that can be achieved. Formally, we give the algorithm in Algorithm 1.

The algorithm terminates after $O(k \cdot \ell^2 \cdot T_f)$ iterations due to the nested loops. We prove by induction that R is a solution to the Join over the arrays $R_1, \ldots, R_i$ after each iteration i. The first iteration simply tries all allowed combinations of the arrays R_1, R_2 and tracks the minimum value for each index, this matches exactly our definition of the Join. Now assume the statement holds for any fixed i. Observe that we only update a value $R[x]$ if there is a respective $M \in A_x$, so none of the values is too small. To show that no value is too large, take any $x \in [\ell_2]$ and let a be the actual minimum value that can be obtained for $R[x]$ in this iteration. Let $j_1, j_2, \ldots, j_{i+1}$ with $x \in \hat{f}(j_1, j_2, \ldots, j_{i+1})$ be the indices that obtain a. Then, there is $y \in [\ell_2]$ such that after joining the first i arrays the value at index y is $a - R_{i+1}[j_{i+1}]$ and $y \in \hat{f}(j_1, j_2, \ldots, j_i)$. This implies $R[y] \leq$

Algorithm 1: JOIN

Input: $R_1, R_2, \ldots, R_{\ell_1}$ for $\ell_1 \geq 2$ with $R_i \in (\mathbb{N} \cup \{\infty\})^{\ell_2}$ for $0 \leq i < \ell_1$, and a computable function $f\colon [\ell_2] \times [\ell_2] \to 2^{[\ell_2]}$.

Output: $R \in (\mathbb{N} \cup \{\infty\})^{\ell_2}$ such that, for all $x \in [\ell_2]$,
$R[x] = \min_{M \in A_x} \sum_{i=1}^{\ell_1} R_i[M[i]]$ with
$A_x = \{M \in ([\ell_2])^{\ell_1} \mid x \in \hat{f}(M[1], M[2], \ldots, M[\ell_2])\}$,
$\hat{f}(x_1, x_2, \ldots, x_k) = \bigcup_{x \in \hat{f}(x_1, x_2, \ldots, x_{k-1})} f(x, x_k)$, and
$\hat{f}(x_1, x_2) = f(x_1, x_2)$.

```
1 R ← R_1
2 for i ← 2 to ℓ_1 do
3     R' ← R
4     foreach (x_1, x_2) ∈ ([ℓ_2])^2 do
5         foreach x ∈ f(x_1, x_2) do
6             R'[x] ← min(R'[x], R[x_1] + R_i[x_2])
7     R ← R'
```

$a - R_{i+1}$ by our induction hypothesis. Further, as both $x \in \hat{f}(j_1, j_2, \ldots, j_{i+1})$ and $y \in \hat{f}(j_1, j_2, \ldots, j_i)$ we have $x \in f(y, j_{i+1})$. Thus, in this iteration, $R[x]$ is set to at most $R[y] + R_{i+1}[j_{i+1}] \leq a$. With this, all values are set correctly and the proof is complete. ■

Observe that in the case of $f(x_1, x_2) = \{x_1 + x_2\}$, which is relevant to this section, the loop in Lines 4 to 6 computes the $(\min, +)$-convolution of the arrays R and R_i. Simply trying all possible combinations as done in the algorithm has a quadratic running time. This cannot be improved without breaking the MINCONV CONJECTURE, which states there is no algorithm computing the $(\min, +)$-convolution of two arrays of length n in time in $O(n^{2-\varepsilon})$ for every $\varepsilon > 0$ [Cyg+19].

5.2.3 Tracking Algorithm

With the JOIN-subroutine at hand, we are able to build a dynamic program solving FAIR CORRELATION CLUSTERING on forests with two colors in a ratio of

1 : 2. We first describe how to apply the algorithm to trees and then generalize it to work on forests.

In the first phase, for each possible difference between the number of br-components and r-components, we compute the minimum number of cuts to obtain a splitting with that difference. In the second phase, we find the splitting for which the sum of edges cut in the first phase and the number of edges required to turn this splitting into a fair partition is minimal. This sum is the inter-cluster cost of that partition, so by Lemma 3.1 this finds a fair partition with the smallest Correlation Clustering cost.

Splitting the tree. In the first phase, our aim is to compute an array D, such that, for all integers $-n \leq x \leq \frac{n}{3}$, $D[x] \subseteq E$ is a minimum-sized set of edges such that $x = br(T - D[x]) - r(T - D[x])$, where $br(T - D[x])$ and $r(T - D[x])$ are the number of br- and r-components in $T - D[x]$, respectively. To mark the case if no such set exists, we expect $D[x] = \mathbb{N}$ to have an infinitely large entry. We fill the array in a dynamic programming way, by computing an array D_v^h for each vertex v, and every possible *head* $h \in \{\emptyset, r, b, rr, br\}$. Here, $D_v^h[x]$, is a minimum-sized set of edges such that in the subtree T_v rooted at v upon removal we have exactly x more br-components than r-components. The head h refers to the colors in the topmost component, which is of particular interest as it might later contain vertices from outside T_v as well. Head $h = r$ refers to a component with a red vertex, $h = br$ with a blue and a red vertex so on. This component is empty ($h = \emptyset$) if the edge above v is cut. The head is not counted as an br-component or r-component for the computation of x. Figure 5.4 gives examples of how a head is composed from the splittings of the children.

In the following, we only show how to compute $\Delta_v^h[x] = |D_v^h[x]|$, the size of the set of edges to obtain a respective splitting. The set $D_v^h[x]$ is, however, obtained by a simple backtracking approach in the same asymptotic running time. If $D_v^h[x] = \mathbb{N}$, we have $\Delta_v^h[x] = \infty$. We initialize all values with $\Delta_v^h[x] = \infty$, meaning we know of no set of edges which upon removal give that head and that difference between br- and r-components. Then, for every red leaf v we set $\Delta_v^r[0] = 0$ and $\Delta_v^\emptyset[-1] = 1$. For every blue leaf v we set $\Delta_v^b[0] = 0$ and $\Delta_v^\emptyset[0] = 1$. This concludes the computations for the leaves, as the only possibilities are to cut the edge above the leaf or not. Now suppose we have finished the computation for all children $u_1, u_2, \ldots, u_k$ of some vertex v. Observe that at most two children

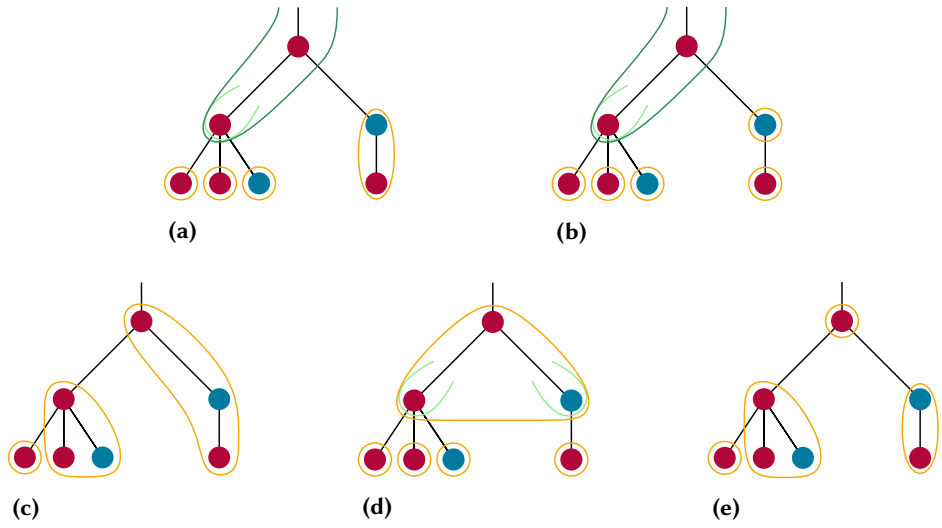

Figure 5.4: Exemplary subtree with various possibilities to obtain a head. Figures 5.4 (a) and 5.4 (b) show splittings with an rr-head (dark green). The choice for the heads of the children (light green) is unambiguous as the only way to obtain an rr-head is to choose the r-head for the left child and an $\emptyset$-head for the right one. Both the left and the right variants have to be considered as they differ in the number of br-components minus the number of r-components. The splittings in Figures 5.4 (c) to 5.4 (e) create an $\emptyset$-head, as they cut the edge above the root of the subtree, so no vertices of the subtree can be part of a component with vertices outside the subtree. Out of these 3 splittings, however, only Figures 5.4 (c) and 5.4 (d) will be further considered as Figure 5.4 (e) obtains the same difference between br- and r-components as Figure 5.4 (c) but cuts one more edge. We note that other splittings obtain an $\emptyset$-head as well that are not listed here.

of v are placed in a head with v. For every head $h \in \{\emptyset, r, b, rr, br\}$ that is formable at vertex v, we try all possibilities to obtain that head.

If $h \in \{r, b\}$ and $c(v)$ corresponds to h, this is done by choosing $\emptyset$ heads for all children. There is no unique splitting of the subtrees however, as for each subtree rooted at some child vertex u_i there is a whole array $D^{\emptyset}_{u_i}$ of possible splittings with different numbers of br- and r-components. To find the best choices for all child vertices, we employ the Join-subroutine that, when called with $f(x_1, x_2) = \{x_1 + x_2\}$ and a list of arrays, returns an array R such that, for all indices x $R[x]$ is the minimum value obtained by summing up exactly one value from each of the input arrays such that the indices of the chosen values sum up to i. We hence set $\Delta^h_v = \textsc{Join}(\Delta^{\emptyset}_{u_1}, \ldots, \Delta^{\emptyset}_{u_k})$. Here and in the following, we only call the Join-subroutine with at least two arrays. If we would only input a single array, we go on as if the Join-subroutine returned that array. We note that here our indexing ranges from $-n$ to $\frac{n}{3}$ while the Join-subroutine assumes positive indices. We hence implicitly assume that an index of x here maps to an index $x + n + 1$ in the subroutine.

If $h = br$ or both $h = rr$ and $c(v)$ corresponds to r, then the heads for all children should be $\emptyset$ except for one child that we place in the same component as v. It then has a head $h' \in \{r, b\}$, depending on h and $c(v)$. We have $h' = r$ if $h = rr$ and $c(v)$ corresponds to R or $h = rb$ and $c(v)$ corresponds to b. Otherwise, $h' = b$. For all $i \in [k]$, we compute an array $\Delta'_{u_i} = \textsc{Join}(\Delta^{\emptyset}_{u_1}, \ldots, \Delta^{\emptyset}_{u_{i-1}}, \Delta^{h'}_{u_i}, \Delta^{\emptyset}_{u_{i+1}}, \ldots, \Delta^{\emptyset}_{u_k})$, referring to u_i having the non-empty head. Lastly, for all $-n \leq x \leq \frac{n}{3}$, we set $\Delta^h_v[x] = \min_{i \in [k]} \Delta'_{u_i}[x]$.

If $h = \emptyset$, then we have to try out all different possibilities for the component v is in and, in each case, cut the edge above v. First assume we want to place v in a brr-component. Then it has to be merged with to vertices, either by taking a head $h' \in \{br, rr\}$ at one child or by taking heads $h_1, h_2 \in \{r, b\}$ at two children. The exact choices for h', h_1, h_2 of course depend on $c(v)$. We compute an array $\Delta_{h'} = \textsc{Join}(\Delta^{\emptyset}_{u_1}, \ldots, \Delta^{\emptyset}_{u_{i-1}}, \Delta^{h'}_{u_i}, \Delta^{\emptyset}_{u_{i+1}}, \ldots, \Delta^{\emptyset}_{u_k})$ for the first option. For the second option we compute arrays
$\Delta_{i,j} = Join(\Delta^{\emptyset}_{u_1}, \ldots, \Delta^{\emptyset}_{u_{i-1}}, \Delta^{h_1}_{u_i}, \Delta^{\emptyset}_{u_{i+1}}, \ldots, \Delta^{\emptyset}_{u_{j-1}}, \Delta^{h_2}_{u_j}, \Delta^{\emptyset}_{u_{j+1}}, \ldots, \Delta^{\emptyset}_{u_k})$ for all pairs of children u_i, u_j of v such that $i < j$ and $\{v, u_i, u_j\}$ is a brr-component. We now have stored the minimum number of cuts for all ways to form a brr-component with v and for all possibilities for x in the arrays $\Delta_{h'}$ and $\Delta_{i,j}$ for all possibilities of i, j. However, v may also be in an r-, b-, rr-, or br-component. Hence, when computing $\Delta^{\emptyset}_v[x]$ we take the minimum value at position x not only among

the arrays $\Delta_{h'}$ and $\Delta_{i,j}$ but also of the arrays $\Delta_v^r, \Delta_v^{br}, \Delta_v^{rr}$, and Δ_v^{br}. Note that here we have to shift all values in Δ_v^r to the left by one since by isolating v we create another r-component. An entry we have written into $\Delta_v^r[x]$ hence should actually be placed in $\Delta_v^r[x-1]$. Similarly, we have to shift Δ_v^{br} to the right, since here we create a new br-component at the top of the subtree. Lastly, as long as v is not the root of T, we have to increase all values in $\Delta_v^\emptyset$ by one, reflecting the extra cut we have to make above v.

After all computations are completed by the correctness of the Join-subroutine and an inductive argument, Δ_v^h is correctly computed for all vertices v and heads h. Note that in the Join-subroutine, as $f(x_1, x_2)$ returns the correct index for merging two subtrees, $\hat{f}(x_1, x_2, \ldots, x_k)$ gives the correct index of merging k subtrees. In particular, $\Delta_r^\emptyset$ is the array containing for each $-n \leq x \leq \frac{n}{3}$ the minimum number of edges to cut such that the there are exactly x more br-components than r-components, where r is the root of T. By adjusting the Join-subroutine to track the exact combination that leads to the minimum value at each position, we also obtain an array D that contains not only the numbers of edges but the sets of edges one has to cut or is marked with $\mathbb{N}$ if no such set exists.

At each node, computing the arrays takes time $O(n^5)$, which is dominated by computing $O(n^2)$ arrays $D_{u,w}$ in time $O(n^3)$ each by Lemma 5.4 since $\ell_1, \ell_2 \in O(n)$. This phase hence takes time in $O(n^6)$.

Assembling a fair clustering. Let D be the set computed in the first phase. Note that each set of edges $D[x]$ directly gives a splitting, namely the partition induced by the connected components in $T - D[x]$.

By Lemma 5.3, the cheapest way to turn the splitting given by $D[x]$ into a clustering of sets of 1 blue and 2 red vertices is found in linear time and incurs $\frac{\max 0, x}{2}$ more cuts. Hence, we find the $-n \leq x \leq \frac{n}{3}$ for which $|D[x]| + \max 0, \frac{x}{2}$ is minimal. We return the corresponding clustering as it has the minimum inter-cluster cost.

This phase takes only constant time per splitting if we tracked the number of components of each type in the first phase and is therefore dominated by the first phase.

Forests. Our algorithm is easily generalized to also solve Fair Correlation Clustering on unconnected forests with two colors in a ratio of $1:2$ by slightly

adapting the first phase. We run the dynamic program as described above for each individual tree. This still takes overall time in $O(n^6)$. For each tree T_i in the forest and every $h \in \{\emptyset, r, b, rr, br\}$, let then $\Delta_{T_i}^{\emptyset}$ denote the array $\Delta_r^{\emptyset}$ with r being the root of tree T_i. To find a splitting of the whole forest and not just on the individual trees, we perform an additional run of the Join-subroutine using these arrays Δ_{T_i} and the function $f(x_1, x_2) = \{x_1 + x_2\}$. This gives us an array R such that $R[x]$ is the minimum number of cuts required to obtain a splitting with exactly x more br-components than r-components for the whole tree rather than for the individual trees. Note that we choose the $\emptyset$-head at each tree as the trees are not connected to each other, so in order to find a splitting we do not yet have to consider how components of different trees are merged, this is done in the second phase. The first phase then outputs an array D that contains the set of edges corresponding to R, which is obtained by a backtracking approach. As the additional subroutine call takes time in $O(n^3)$, the asymptotic run time of the algorithm does not change. This gives the following result.

▶ **Theorem 5.5.** Fair Correlation Clustering on forests with two colors in a ratio of $1 : 2$ can be solved in time in $O(n^6)$. ◀

5.3 Small Clusters

To obtain an algorithm that handles more colors and different color ratios, we generalize our approach for the $1 : 2$ color ratio case from the previous section. We achieve the following result.

▶ **Theorem 5.6.** Let F be a forest of n vertices, each colored in one of $k \geq 2$ colors. Let the colors be distributed in a ratio of $c_1 : c_2 : \ldots : c_k$ with $c_i \in \mathbb{N}_{>0}$ for all $i \in [k]$ and $\gcd(c_1, c_2, \ldots, c_k) = 1$. Then Fair Correlation Clustering on F can be solved in time in $O(n^{2\text{setvars}+\text{setmax}+2} \cdot \text{setvars}^{\text{setmax}})$, where $\text{setvars} = \prod_{i=1}^{k}(c_i + 1)$ and $\text{setmax} = \sum_{i=1}^{k} c_i$. ◀

Once more, the algorithm runs in two phases. First, it creates a list of possible splittings, i.e., partitions in which, for every color, every component has at most as many vertices of that color as a minimum-sized fair component has. In the second phase, it checks for these splittings whether they can be merged into a fair clustering. Among these, it returns the one of minimum cost. We first give the algorithm solving the problem on trees and then generalize it to also capture forests.

Splitting the forest. For the first phase in the 1:2 approach, we employed a dynamic program that kept track of the minimum number to obtain a splitting for each possible cost incurred by the reassembling in the second phase. Unfortunately, if we are given a graph with $k \geq 2$ colors in a ratio of $c_1 : c_2 : \ldots : c_k$, then the number of cuts that are required in the second phase is not always as easily bounded by the difference of the number of two component types like r- and br-components in the $1 : 2$ case. However, we find that it suffices to track the minimum number of cuts required to obtain any possible coloring of a splitting.

We first bound the number of possible colorings of a splitting. As during the dynamic program we consider splittings of a subgraph of G most of the time, we also have to count all possible colorings of splittings of less than n vertices.

▶ **Lemma 5.7.** Let U be a set of n elements, colored in $k \in \mathbb{N}_{>1}$ colors, and let $d_1, d_2, \ldots, d_k \in \mathbb{N}$. Let $\mathcal{S}$ be the set of all possible partitions of subsets of U such that for every color i there are at most d_i vertices of that color in each cluster. Let C be the set of all colorings of partitions in $\mathcal{S}$. Then, $|C| \leq (n+1)^{\text{setvars}-1}$, where setvars $= \prod_{i=1}^{k}(d_i + 1)$. ◀

Proof. The number of sets with different colorings is at most setvars as there are 0 to d_i many vertices of color i in each component. Thus, a coloring of a partition $\mathcal{P}$ using only these sets is characterized by an array of size setvars with values in $[n] \cup \{0\}$ as no component occurs more than n times. There are $(n+1)^{\text{setvars}}$ ways to fill such an array. However, as the set colorings together have to form a partition, the last entry is determined by the first setvars $-$ 1 entries, giving only $(n+1)^{\text{setvars}-1}$ possibilities. ■

With this, we now employ a dynamic program similar to the one presented in Section 5.2 but track the minimum cut cost for all colorings of splittings. It is given by the following lemma.

▶ **Lemma 5.8.** Let $F = (V, E)$ be a forest with vertices in k colors. Further, let $d_1, d_2, \ldots, d_k \in \mathbb{N}$ and $\mathcal{S}$ be the set of all possible partitions of V such that there are at most d_i vertices of color i in each cluster for $i \in [k]$. Let C be the set of all colorings of partitions in $\mathcal{S}$. Then, in time in $O\big(n^{2\text{setvars}+\text{setmax}+2} \cdot \text{setvars}^{\text{setmax}}\big)$ with setvars $= \prod_{i=1}^{k}(d_i + 1)$ and setmax $= \sum_{i=1}^{k} d_i$, for all $C \in C$, we find a minimum-sized set $D_C \subseteq E$ such that the connected components in $F - D_C$ form a partition of the vertices with coloring C or state that there is no such set. ◀

Proof. We first describe how to solve the problem on a tree T and then generalize the approach to forests. We call a partition of the vertices such that for every color i there are at most d_i vertices of that color in each cluster a *splitting*.

We employ a dynamic program that computes the set D_C for the colorings of all possible splittings and all subtrees rooted at each vertex in T. We do so iteratively, by starting to compute all possible splittings at the leaves and augmenting them towards the root. Thereby, the connected component that is connected to the parent of the current subtree's root is of particular importance as it is the only connected component that can be augmented by vertices outside the subtree. We call this component the *head*. Note that the head is empty if the edge between the root and its parent is cut. We do not count the head in the coloring of the splitting and only give it explicitly. Formally, for every $v \in V$, every possible coloring of a splitting C, and every possible coloring h of the head we compute $D_v^h[C] \subseteq E$, the minimum-sized set of edges such that the connected components of $T_v - D_v^h[C]$ form a splitting with coloring C and head h. We set $D_v^h[C] = \mathbb{N}$, an infinitely large set, if no such set exists.

Let all $D_v^h[C]$ be initialized with $\mathbb{N}$. Then, for every leaf v with parent w, we set $D_v^{h_{c(v)}}[C_\emptyset] = \emptyset$, where $h_{c(v)}$ is the coloring of the component $\{v\}$ and $C_\emptyset$ the coloring of the partition over the empty set. Also, we set $D_v^{h_\emptyset}[C_{c(v)}] = \{\{v, w\}\}$, where the vertex v is not placed in the head as the edge to its parent is cut. As to cut or not to cut the edge above are the only options for leaves, this part of the array is now completed.

Next, suppose we have finished the computation for all children of some vertex v. For every possible coloring h of the head that is formable at vertex v, we try all possibilities to obtain that coloring.

To this end, first assume h to be non-empty. Therefore, v has to be placed in the head. Let $h_{-c(v)}$ denote the coloring obtained by decreasing h by one at color $c(v)$. To obtain head h, we hence have to choose colorings of splittings of the subtrees rooted at the children $u_1, u_2, \ldots, u_\ell$ of v such that their respective heads $h_{u_1}, h_{u_2}, \ldots, h_{u_\ell}$ combine to $h_{-c(v)}$. A *combination* of colorings $C_1, C_2, \ldots, C_\ell$ refers to the coloring of the union of partitions $M_1, M_2, \ldots, M_\ell$ that have the respective colorings and is defined as the element-wise sum over the arrays $C_1, C_2, \ldots, C_\ell$. Often, there are multiple ways to choose heads for the child vertices that fulfill this requirement. As every head is of size at most setmax, $h_{-c(v)}$ and contains v, it is composed of less than setmax non-empty heads. As there are at most setvars possible heads and we have to choose less than setmax children, there

are at most $\binom{n}{\text{setmax}-1} \cdot \text{setvars}^{\text{setmax}-1} < n^{\text{setmax}-1} \cdot \text{setvars}^{\text{setmax}-1}$ possible ways to form $h_{-c(v)}$ with the children of v. Let each way be described by a function H assigning each child of v a certain, possibly empty, head. Then, even for a fixed H, there are multiple splittings possible. This stems from the fact that even if the head $H(u)$ for a child u is fixed, there might be multiple splittings of the subtree of u with different colorings resulting in that head. For each possible H, we hence employ the Join-subroutine with the arrays $D_u^{H(u)}$ for all children u using the cardinality of the sets as input for the subroutine. For the sake of readability, we index the arrays here by some vector C instead of a single numerical index as used in the algorithmic description of the Join-subroutine. We implicitly assume that each possible coloring is represented by a positive integer. By letting these indices enumerate the vectors in a structured way, converting between the two formats only costs an additional time factor in $O(n)$.

For $f(x_1, x_2)$ we give the function returning a set containing only the index of the coloring obtained by combining the colorings indexed by x_1 and x_2, which is computable in time in $O(n)$. Combining the colorings means for each set coloring summing the occurrences in both partition colorings. Thereby, $\hat{f}(x_1, x_2, \ldots, x_k)$ as defined in the Join-subroutine returns the index of the combination of the colorings indexed by $x_1, x_2, \ldots, x_k$. Note that there are at most n arrays and each is of length less than $(n+1)^{\text{setvars}-1}$ as there are so many different colorings by Lemma 5.7. After executing the Join-subroutine, by Lemma 5.4, we obtain an array D_H that contains the minimum cut cost required for all possible colorings that can be achieved by splitting according to H. By modifying the Join-subroutine slightly to use a simple backtracking approach, we also obtain the set $D \subseteq E$ that achieves this cut cost. We conclude our computation of D_v^h by element-wisely taking the minimum-sized set over all computed arrays D_H for the possible assignments H.

If h is the empty head, i.e., the edge above v is cut, then v is placed in a component that is either of size setmax or has a coloring corresponding to some head h'. In the first case, we compute an array D_{full} in the same manner as described above by trying all suitable assignments H and employing the Join-subroutine. In the second case, we simply take the already filled array $D_v^{h'}$. Note that in both cases we have to increment all values in the array by one to reflect cutting the edge above v, except if v is the root vertex. Also, we have to move the values in the arrays around, in order to reflect that the component containing v is no longer a head but with the edge above v cut should also be

counted in the coloring of the splitting. Hence, the entry $D_{\text{full}}[C]$ is actually stored at $D_{\text{full}}[C_{-\text{full}}]$ with $C_{-\text{full}}$ being the coloring C minus the coloring of a minimum-sized fair cluster. If no such entry $D_{\text{full}}[C_{-\text{full}}]$ exists, we assume it to be ∞. The same goes for accessing the arrays $D_v^{h'}$ where we have to subtract the coloring h' from the index. Taking the element-wise minimum-sized element over the such modified arrays D_{full} and $D_v^{h'}$ for all possibilities for h' yields $D_v^{\emptyset}$.

By the correctness of the Join-subroutine and as we try out all possibilities to build the specified heads and colorings at every vertex, we thus know that after completing the computation at the root r of T, the array $D_r^{\emptyset}$ contains for every possible coloring of a splitting of the tree the minimum cut cost to achieve that coloring.

For each of the n vertices and the setvars possible heads, we call the Join-subroutine at most $n^{\text{setmax}-1} \cdot \text{setvars}^{\text{setmax}-1}$ many times. Each time, we call it with at most n arrays and, as by Lemma 5.7 there are $O(n^{\text{setmax}})$ possible colorings, all these arrays have that many elements. Hence, each subroutine call takes time in $O\left(n \cdot \left(n^{\text{setvars}}\right)^2\right) = O(n^{2\text{setvars}+1})$, so the algorithm takes time in $O(n^{2\text{setvars}+\text{setmax}+2} \cdot \text{setvars}^{\text{setmax}})$, including an additional factor in $O(n)$ to account for converting the indices for the Join-subroutine.

When the input graph is not a tree but a forest F, we apply the dynamic program on every tree in the forest. Then, we additionally run the Join-subroutine with the arrays for the $\emptyset$-head at the roots of all trees in the forest. The resulting array contains all minimum-cost solutions from all possible combinations from colorings of splittings from the individual trees and is returned as output. The one additional subroutine does not change the asymptotic running time. ■

Because of Lemmas 3.2 and 3.3 it suffices to consider partitions as possible solutions that have at most c_i vertices of color i in each cluster, for all $i \in [k]$. We hence apply Lemma 5.8 on the forest F and set $d_i = c_i$ for all $i \in [k]$. This way, for every possible coloring of a splitting we find the minimum set of edges to obtain a splitting with that coloring.

Assembling a fair clustering. Let D be the array produced in the first phase, i.e., for every coloring C of a splitting, $D[C]$ is a minimum-sized set of edges such that the connected components in $F - D[C]$ induce a partition with coloring C. In the second phase, we have to find the splitting that gives the minimum Correlation Clustering cost. We do so by deciding for each splitting whether

it is *assemblable*, i.e., whether its clusters can be merged such that it becomes a fair solution with all clusters being no larger than setmax. Among these, we return the one with the minimum inter-cluster cost computed in the first phase.

This suffices because of the following reasons. First, note that deciding assemblability only depends on the coloring of the splitting so it does not hurt that in the first phase we tracked only all possible colorings of splittings and not all possible splittings themselves. Second, we do not have to consider further edge cuts in this phase: Assume we have a splitting S with coloring C_S and we would obtain a better cost by further cutting a edges in S, obtaining another splitting S' of coloring $C_{S'}$. However, as we filled the array D correctly, there is an entry $D[C_{S'}]$ and $|D[C_{S'}]| \leq |D[C_S]| + a$. As we will consider this value in finding the minimum anyway, there is no need to think about cutting the splittings any further. Third, the minimum inter-cluster cost yields the minimum Correlation Clustering cost by Lemma 3.1. When merging clusters, the inter-cluster cost computed in the first phase may decrease but not increase. If it decreases, we overestimate the cost. However, this case implies that there is an edge between the two clusters and as they are still of size at most setmax when merged, in the first phase we will also have found another splitting considering this case.

We employ a dynamic program to decide the assemblability for all possible $O(n^{\text{setvars}})$ colorings of splittings. Let the size of a partition coloring be the number of set colorings in that partition coloring (not necessarily the number of different set colorings). We decide assemblability for all possible colorings of splittings from smallest to largest. Note that each such coloring is of size at least $\frac{n}{\text{setmax}}$. If it is of size exactly $\frac{n}{\text{setmax}}$, then all contained set colorings are of size setmax, so this partition coloring is assemblable if and only if all set colorings are fair. Now assume we have found all assemblable colorings of splittings of size exactly $j \geq \frac{n}{\text{setmax}}$. Assume a partition coloring C of size $j+1$ is assemblable. Then, at least two set colorings C_1, C_2 from C are merged together. Hence, let C' be the partition coloring obtained by removing the set colorings C_1, C_2 from C and adding the set coloring of the combined coloring of C_1 and C_2. Now, C' is of size j and is assemblable. Thus, every assemblable splitting with $j+1$ components has an assemblable splitting with j components. The other way round, if we split a set coloring of an assemblable partition coloring of size j we obtain an assemblable partition coloring of size $j+1$. Hence, we find all assemblable colorings of splittings of size $j+1$ by for each assemblable partition coloring of size j (less than n^{setvars} many) trying each possible way to split one of

its set colorings (less than $i \cdot 2^{\text{setmax}}$ as there are j set colorings each of size at most setmax). Thus, to compute all assemblable colorings of splittings of size $j+1$, we need time in $O\big(n^{\text{setvars}} \cdot j \cdot 2^{\text{setmax}}\big)$, which implies a total time for the $n - \frac{n}{\text{setmax}}$ iterations in the second phase in $O\big(n^{\text{setvars}+2} \cdot 2^{\text{setmax}}\big)$. This is dominated by the running time of the first phase. The complete algorithm hence runs in time in $O\big(n^{2\text{setvars}+\text{setmax}+2} \cdot \text{setvars}^{\text{setmax}}\big)$, which yields Theorem 5.6.

This gives an algorithm that solves Fair Correlation Clustering on arbitrary forests. The running time however may be exponential in the number of vertices depending on the color ratio in the forest.

5.4 Few Clusters

The algorithm presented in the previous section runs in polynomial time if the colors in the graph are distributed in a way such that each cluster in a minimum-cost solution is of constant size. The worst running time is obtained when there are very large but few clusters. For this case, we offer another algorithm, which runs in polynomial time if the number of clusters is constant. However, it is limited to instances where the forest is colored in two colors in a ratio of $1 : c$ for some $c \in \mathbb{N}$.

The algorithm uses a subroutine that computes the minimum number of cuts that are required to slice off clusters of specific sizes from the tree. It is given by Lemma 5.9.

▶ **Lemma 5.9.** Let $T = (V, E)$ be a tree rooted at $r \in V$ and $k \in \mathbb{N}$. Then, we can compute an array R such that, for each $a_0 \in [n]$ and $a = a_1, a_2, \ldots, a_k \in ([n-1] \cup \{0\})^k$ with $a_i \geq a_{i+1}$ for $i \in [k-1]$ and $\sum_{i=0}^{k} a_i = n$, we have that $R[a_0, a]$ is the partition $\mathcal{P} = \{S_0, S_1, \ldots, S_k\}$ of V with minimum inter-cluster cost that suffices $r \in S_0$ and $|S_i| = a_i$ for $i \in [k]$. The computation time is in $O\big((k+3)! \cdot n^{2k+3}\big)$. ◀

Proof. We give a construction such that $R[a_0, a]$ stores not the partition itself but the incurred inter-cluster cost. By a simple backtracking approach, the partitions are obtained as well.

We employ a dynamic program that involves using the Join-subroutine. For the sake of readability, we index the arrays here by some vector $a \in [n]^k$ and $a_0 \in [n]$ instead of a single numerical index as used in the algorithmic description of the Join-subroutine. We implicitly assume that each possible a_0, a

is represented by some index in $[n^{k+1}]$. By letting these indices enumerate the vectors in a structured way, converting between the two formats only costs an additional time factor in $O(k)$.

Starting at the leaves and continuing at the vertices for which all children have finished their computation, we compute an array R_v with the properties described for R but for the subtree T_v for each vertex $v \in V$. In particular, for every vertex v we do the following. Let R_v^0 be an array with ∞-values at all indices except for $R_v^0[1, (0, 0, \ldots, 0)] = 0$, as this is the only possible entry for the tree $T[\{v\}]$.

If v has no children, then $R = R_v^0$. Otherwise, let the children of v be $u_1, u_2, \ldots, u_\ell$. Then we call the Join-subroutine with the arrays $R_v^0, R_{u_1}, R_{u_2}, \ldots, R_{u_\ell}$. We have to define f such that it gives all possibilities to combine the children's subtrees partitions and v. For all possible values of a_0, a and a_0', a' recall that $f((a_0, a), (a_0', a'))$ should return a set of indices of the form (a_0'', a''). Each such index describes a combination of all possibilities for v and the already considered children (a_0, a) and the possibilities for the next child (a_0', a'). First, we consider the possibility to cut the edge between v and the child u that is represented by (a_0', a''). Then, we add all possible ways of merging the two sets with their $k+1$ clusters each. As we cut the edge $\{u, v\}$, there are k possible ways to place the cluster containing u (all but the cluster containing v) and then there are $k!$ ways to assign the remaining clusters. All these are put into the set $f((a_0, a), (a_0', a'))$. Second, we assume the edge $\{u, v\}$ is not cut. Then, the clusters containing v and u have to be merged, so there are only $k!$ possible ways to assign the other clusters. In particular, for all indices (a_0'', a'') put into $f((a_0, a), (a_0', a'))$ this way, we have $a_0'' = a_0 + a_0'$. Note that f can be computed in $O(k \cdot k!)$. Note that $\hat{f}(x_1, x_2, \ldots, x_\ell)$ as defined in the Join-subroutine lists all possibilities to cut the combined tree as it iteratively combines all possibilities for the first child and the vertex v and for the resulting tree lists all possible combinations with the next child and so on. The Join-subroutine takes time in $O\left((k+1) \cdot \left(n^{k+1}\right)^2 \cdot (k \cdot k!) \cdot k\right)$, which is in $O\left((k+3)! \cdot n^{2k+2}\right)$. All $O(n)$ calls of the subroutine hence take time in $O\left((k+3)! \cdot n^{2k+3}\right)$ ■

With this, we are able to give an algorithm for graphs with two colors in a ratio of $1 : c$, which runs in polynomial time if there is only a constant number of clusters, i.e., if $c \in \Theta(n)$.

▶ **Theorem 5.10.** Let F be an n-vertex forest with two colors in a ratio of $1 : c$ with $c \in \mathbb{N}_{>0}$ and let $p = \frac{n}{c+1}$. Then, Fair Correlation Clustering on F can be solved in $O\left(n^{p^3+p^2+p}\right)$. ◀

Proof. Note that, if there are c red vertices per 1 blue vertex, $p = \frac{n}{c+1}$ is the number of blue vertices. By Lemma 3.2, any minimum-cost clustering consists of p clusters, each containing exactly one blue vertex, and from Lemma 3.1 we know that it suffices to minimize the number of edges cut by any such clustering. All blue vertices are to be placed in separate clusters. They are separated by cutting at most $p-1$ edges, so we try all of the $O\left((p-1)\cdot\binom{n-1}{p-1}\right)$ subsets of edges of size at most $p-1$. Having cut these edges, we have ℓ trees $T_1, T_2, \ldots, T_\ell$, with p of them containing exactly one blue vertex and the others no blue vertices. We root the trees at the blue vertex if they have one or at an arbitrary vertex otherwise. For each tree T_i, let r_i be the number of red vertices. If we have exactly p trees and $r_i = c$ for all $i \in [p]$, we have found a minimum-cost clustering, where the i-th cluster is simply the set of vertices of T_i for all $i \in [p]$. Otherwise, we must cut off parts of the trees and assign them to other clusters in order to make the partition fair. To this end, for each tree T_i we compute an array R_i that states the cost of cutting up to $p-1$ parts of certain sizes off. More precisely, $R_i[(a_1, a_2, \ldots, a_{p-1})]$ is the number of cuts required to cut off $p-1$ clusters of size $a_1, a_2, \ldots, a_{p-1}$, respectively, and ∞ if there is no such way as $\sum_{i=1}^{p-1} > r_i$. It suffices to compute $R_i[(a_1, a_2, \ldots, a_{p-1})]$ with $0 \leq a_i \leq a_{i+1} \leq n$ for $i \in [p-2]$.

We compute these arrays employing Lemma 5.9. Note that here we omitted the a_0 used in the lemma, which here refers to the number of vertices *not* cut from the tree. However, a_0 is still unambiguously defined over a as all the values sum up to the number of vertices in this tree. Further, by connecting all trees without blue vertices to some newly added auxiliary vertex z and using this tree rooted at z as input to Lemma 5.9, we reduce the number of subroutine calls to $p+1$. Then, the only entries from the array obtained for the all-red tree we consider are the ones with $a_0 = 1$ as we do not want to merge z in a cluster but every vertex except z from this tree has to be merged into another cluster. We call the array obtained from this tree R_0 and the arrays obtained for the other trees $R_1, R_2, \ldots, R_p$, respectively.

Note that every fair clustering is characterized by choosing one entry from each array R_i and assigning the cut-off parts to other clusters. As each array has less than $\frac{n^p}{p!}$ entries and there are at most $(p!)^p$ ways to assign the cut-off

parts to clusters, there are at most n^{p^2} possibilities in total. For each of these, we compute in linear time whether they result in a fair clustering. Among these fair clusterings, we return the one with the minimum inter-cluster cost, computed by taking the sum over the chosen entries from the arrays R_i. By Lemma 3.1, this clustering has the minimum Correlation Clustering cost. We obtain a total running time of $O\left((p-1)\cdot\binom{n-1}{p-1}\cdot\left((p+1)\cdot\left(n^{p+3}+n^{p^2+p-2}\right)+n^{p^2+1}\right)\right)\subseteq O\left(n^{p^3+p^2+p}\right)$. ■

Combining the results of Theorems 5.6 and 5.10, we see that for the case of a forest with two colors in a ratio of $1:c$ for some $c\in\mathbb{N}_{>0}$, there are polynomial time algorithms when the clusters are either of constant size or size in $\Theta(n)$. As Theorem 4.1 states that Fair Correlation Clustering on forests is NP-hard, we hence know that this hardness evolves somewhere between the two extremes.

6 Relaxed Fairness

It might look like the hardness results for Fair Correlation Clustering are due to the very strict definition of fairness, which enforces clusters of a specific size on forests. However, in this chapter, we prove that even when relaxing the fairness requirements our results essentially still hold.

6.1 Definitions

We use the relaxed fairness constraint as proposed by Bera et al. [Ber+19] and employed for Fair Correlation Clustering by Ahmadi et al. [Ahm+20a]. For the following definitions, given a set U colored by a function $c : U \to k$, by $U_i = \{u \in U \mid c(u) = i\}$ we denote the set of vertices of color i for all $i \in [k]$.

▶ **Definition 6.1 (Relaxed Fair Set).** Let U be a finite set of elements colored by a function $c : U \to [k]$ for some $k \in \mathbb{N}_{>0}$ and let $p_i, q_i \in \mathbb{Q}$ with $0 < p_i \leq \frac{|U_i|}{|U|} \leq q_i < 1$ for all $i \in [k]$. Then, some $S \subseteq U$ is relaxed fair with regard to the q_i and p_i if and only if for all colors $i \in [k]$ we have $p_i \leq \frac{|S \cap U_i|}{|S|} \leq q_i$. ◀

Note that we require p_i and q_i to be such that an exact fair solution is also relaxed fair. Further, we exclude setting p_i or q_i to 0 as this would allow clusters that do not include every color, which we do not consider fair.

▶ **Definition 6.2 (Relaxed Fair Partition).** Let U be a finite set of elements colored by a function $c : U \to [k]$ for some $k \in \mathbb{N}_{>0}$ and let $p_i, q_i \in \mathbb{Q}$ with $0 < p_i \leq \frac{|U_i|}{|U|} \leq q_i < 1$ for all $i \in [k]$. Then, a partition $S_1 \cup S_2 \cup \ldots \cup S_\ell = U$ is relaxed fair with regard to the q_i and p_i if and only if all sets $S_1, S_2, \ldots, S_\ell$ are relaxed fair with regard to the q_i and p_i. ◀

Relaxed Fair Correlation Clustering

Input: Graph $G = (V, E)$, coloring $c\colon V \to [k]$, $p_i, q_i \in \mathbb{Q}$ with $0 < p_i \leq \frac{|U_i|}{|U|} \leq q_i < 1$ for all $i \in [k]$.

Task: Find a relaxed fair partition $\mathcal{P}$ of V with regard to the p_i and q_i that minimizes cost($\mathcal{P}$).

While we use the above definition for our hardness results, we restrict the possibilities for the p_i and q_i for our algorithms.

▶ **Definition 6.3 (α-relaxed Fair Set).** Let U be a finite set of elements colored by a function $c : U \to [k]$ for some $k \in \mathbb{N}_{>0}$ and let $0 < \alpha < 1$. Then, some $S \subseteq U$ is α-relaxed fair if and only if it is relaxed fair with regard to $p_i = \frac{\alpha|U_i|}{|U|}$ and $q_i = \frac{|U_i|}{\alpha|U|}$ for all $i \in [k]$. ◀

▶ **Definition 6.4 (α-relaxed Fair Partition).** Let U be a finite set of elements colored by a function $c : U \to [k]$ for some $k \in \mathbb{N}_{>0}$ and let $0 < \alpha < 1$. Then, a partition $S_1 \cup S_2 \cup \ldots \cup S_\ell = U$ is α-relaxed fair if and only if all sets $S_1, S_2, \ldots, S_\ell$ are α-relaxed fair. ◀

α-relaxed Fair Correlation Clustering

Input: Graph $G = (V, E)$, coloring $c\colon V \to [k]$, $0 < \alpha < 1$.

Task: Find a α-relaxed fair partition $\mathcal{P}$ of V that minimizes cost($\mathcal{P}$).

.

6.2 Hardness

The hardness result for exact fairness on paths, see Theorem 4.4, directly carries over to the relaxed fairness setting. This is due to it only considering instances in which there are exactly two vertices of each color. As any relaxed fair clustering still requires at least one vertex of every color in each cluster, this means that every relaxed clustering either consists of a single cluster or two clusters, each with one vertex of every color. Thereby, relaxing fairness makes no difference in these instances.

▶ **Corollary 6.5.** Relaxed Fair Correlation Clusteringon paths is NP-hard, even when limited to instances with exactly 2 vertices of each color. ◀

Our other hardness proofs for relaxed fairness are based on the notion that we can use similar constructions as for exact fairness and additionally prove that in these instances the minimum-cost solution has to be exactly fair and not just relaxed fair. To this end, we require a lemma giving a lower bound on the intra-cluster cost of clusterings.

▶ **Lemma 6.6.** Let $G = (V, E)$ be an n-vertex m-edge graph and $\mathcal{P}$ a partition of V with an inter-cluster cost of χ. Then, the intra-cluster cost of $\mathcal{P}$ is at least $\frac{n^2}{2|\mathcal{P}|} - \frac{n}{2} - m + \chi$. If $|S| = \frac{n}{|\mathcal{P}|}$ for all clusters $S \in \mathcal{P}$, then the intra-cluster cost of $\mathcal{P}$ is $\psi = \frac{n^2}{2|\mathcal{P}|} - \frac{n}{2} - m + \chi$. ◀

Proof. We first prove the lower bound. We employ the Cauchy-Schwarz Inequality in the ℓ-dimensional Euclidean space. It states that for every $\ell \in \mathbb{N}$, $x_1, x_2, \ldots, x_\ell$, and $y_1, y_2, \ldots, y_\ell$, we have

$$\left(\sum_{i=1}^{\ell} x_i y_i\right)^2 \leq \left(\sum_{i=1}^{\ell} x_i^2\right) \cdot \left(\sum_{i=1}^{\ell} y_i^2\right).$$

In particular, with $y_i = 1$ for all $i \in [\ell]$, we have

$$\left(\sum_{i=1}^{\ell} x_i\right)^2 \leq \ell \cdot \sum_{i=1}^{\ell} x_i^2.$$

Observe that we can write the intra-cluster cost ψ of $\mathcal{P}$ as

$$\begin{aligned}
\psi &= \left(\sum_{S \in \mathcal{P}} \frac{|S| \cdot (|S| - 1)}{2}\right) - (m - \chi) \\
&= \frac{1}{2}\left(\sum_{S \in \mathcal{P}} |S|^2\right) - \left(\sum_{S \in \mathcal{P}} \frac{|S|}{2}\right) - m + \chi \\
&= \frac{1}{2}\left(\sum_{S \in \mathcal{P}} |S|^2\right) - \frac{n}{2} - m + \chi.
\end{aligned}$$

By Cauchy-Schwarz, we have

$$\sum_{S\in\mathcal{P}} |S|^2 \geq \frac{1}{|\mathcal{P}|} \cdot \left(\sum_{S\in\mathcal{P}} |S|\right)^2 = \frac{n^2}{|\mathcal{P}|}.$$

This bounds the intra-cluster cost from below by

$$\psi \geq \frac{n^2}{2|\mathcal{P}|} - \frac{n}{2} - m + \chi.$$

For the second statement, assume all clusters of $\mathcal{P}$ to be of size $\frac{n}{|\mathcal{P}|}$. Then, there are $\frac{1}{2} \cdot \frac{n}{|\mathcal{P}|} \cdot \left(\frac{n}{|\mathcal{P}|} - 1\right)$ pairs of vertices in each cluster. Thereby, we have

$$\begin{aligned}\psi &= |\mathcal{P}| \cdot \frac{1}{2} \cdot \frac{n}{|\mathcal{P}|} \cdot \left(\frac{n}{|\mathcal{P}|} - 1\right) - (m - \chi) \\ &= \frac{n^2}{2|\mathcal{P}|} - \frac{n}{2} - m + \chi.\end{aligned}$$

■

We further show that no clustering with clusters of unequal size achieves the lower bound given by Lemma 6.6.

▶ **Lemma 6.7.** Let $G = (V, E)$ be an n-vertex m-edge graph and $\mathcal{P}$ a partition of V with an inter-cluster cost of χ such that there is a cluster $S \in \mathcal{P}$ with $|S| = \frac{n}{|\mathcal{P}|} + a$ for some $a \geq 0$. Then, the intra-cluster cost of $\mathcal{P}$ is $\psi \geq \frac{a^2|\mathcal{P}|}{2|\mathcal{P}|-2} + \frac{n^2}{2|\mathcal{P}|} - \frac{n}{2} - m + \chi$. ◀

Proof. If $a = 0$, the statement is implied by Lemma 6.6. So, assume $a > 0$. We write the intra-cluster cost as

$$\psi = \frac{1}{2} \cdot \left(\frac{n}{|\mathcal{P}|} + a\right) \cdot \left(\frac{n}{|\mathcal{P}|} + a - 1\right) + \psi_{\text{rest}}$$

with ψ_{rest} being the intra-cluster cost incurred by $\mathcal{P} \setminus \{S\}$. By applying Lemma 6.6

on $\mathcal{P} \setminus \{S\}$, we have

$$\begin{aligned}\psi &\geq \frac{1}{2} \cdot \left(\frac{n}{|\mathcal{P}|} + a\right) \cdot \left(\frac{n}{|\mathcal{P}|} + a - 1\right) + \frac{\left(n - \left(\frac{n}{|\mathcal{P}|} + a\right)\right)^2}{2(|\mathcal{P}| - 1)} - \frac{n - \left(\frac{n}{|\mathcal{P}|} + a\right)}{2} - m + \chi \\ &= \frac{n^2}{2|\mathcal{P}|^2} + \frac{an}{|\mathcal{P}|} + \frac{a^2}{2} - \frac{n}{2|\mathcal{P}|} - \frac{a}{2} + \frac{n^2 - 2n^2/|\mathcal{P}| - 2an + n^2/|\mathcal{P}|^2 + 2a\frac{n}{|\mathcal{P}|} + a^2}{2|\mathcal{P}| - 2} \\ &\quad - \frac{n}{2} + \frac{n}{2|\mathcal{P}|} + \frac{a}{2} - m + \chi.\end{aligned}$$

Bringing the first summands to a common denominator of $2|\mathcal{P}| - 2$ yields

$$\begin{aligned}\psi &\geq \left(\frac{n^2(|\mathcal{P}| - 1)}{|\mathcal{P}|^2} + \frac{an(2|\mathcal{P}| - 2)}{|\mathcal{P}|} + a^2(|\mathcal{P}| - 1) + n^2 - \frac{2n^2}{|\mathcal{P}|} - 2an + \frac{n^2}{|\mathcal{P}|^2} + \frac{2an}{|\mathcal{P}|} + a^2\right) \\ &\quad \Big/ (2|\mathcal{P}| - 2) - \frac{n}{2} - m + \chi \\ &= \left(\frac{n^2|\mathcal{P}|}{|\mathcal{P}|^2} + \frac{2an|\mathcal{P}|}{|\mathcal{P}|} + a^2|\mathcal{P}| + n^2 - \frac{2n^2}{|\mathcal{P}|} - 2an\right) \Big/ (2|\mathcal{P}| - 2) - \frac{n}{2} - m + \chi \\ &= \left(-\frac{n^2}{|\mathcal{P}|} + a^2|\mathcal{P}| + n^2\right) \Big/ (2|\mathcal{P}| - 2) - \frac{n}{2} - m + \chi.\end{aligned}$$

We then add $0 = -\frac{n^2}{2|\mathcal{P}|} \cdot \frac{2|P|-2}{2|P|-2} + \frac{n^2}{2|\mathcal{P}|}$ and obtain

$$\begin{aligned}\psi &\geq \left(-\frac{n^2}{|\mathcal{P}|} + a^2|\mathcal{P}| + n^2 - \frac{n^2(|\mathcal{P}| - 1)}{|\mathcal{P}|}\right) \Big/ (2|\mathcal{P}| - 2) + \frac{n^2}{2|\mathcal{P}|} - \frac{n}{2} - m + \chi \\ &= \frac{a^2|\mathcal{P}|}{2|\mathcal{P}| - 2} + \frac{n^2}{2|\mathcal{P}|} - \frac{n}{2} - m + \chi.\end{aligned}$$

■

Observe that as $|\mathcal{P}| > 1$ and $a \neq 0$ this means that such a clustering never achieves the lower bound given by Lemma 6.6. In particular, this means that for fixed inter-cluster costs in minimum-cost fair clusterings in forests all clusters are of equal size. This way, we are able to transfer some hardness results obtained for exact fairness to relaxed fairness.

▶ **Theorem 6.8.** For every choice of $0 < p_1 \leq \frac{1}{c+1} \leq q_1 < 1$ and $0 < p_2 \leq \frac{c}{c+1} \leq q_2 < 1$, Relaxed Fair Correlation Clusteringon forests with two colors in

a ratio of $1 : c$ is NP-hard. It remains NP-hard when arbitrarily restricting the shape of the trees in the forest as long as for every $a \in \mathbb{N}$ it is possible to form a tree with a vertices. ◀

Proof. We reduce from 3-Partition. Recall that there are $3p$ values $a_1, a_2, \ldots, a_{3p}$ and the task is to partition them in triplets that each sum to B. We construct a forest F as follows. For every a_i we construct an arbitrary tree of a_i red vertices. Further, we let there be p isolated blue vertices. Note that the ratio between blue and red vertices is $1 : B$. We now show that there is a relaxed fair clustering $\mathcal{P}$ such that

$$\text{cost}(\mathcal{P}) \leq p \cdot \frac{B(B+1)}{2} - p(B-3)$$

if and only if the given instance is a yes-instance for 3-Partition.

If we have a yes-instance of 3-Partition, then there is a partition of the set of trees into p clusters of size B. By assigning the blue vertices arbitrarily to one unique cluster each, we hence obtain an exactly fair partition, which is thus also relaxed fair. As there are no edges between the clusters and each cluster consists of $B+1$ vertices and $B-3$ edges, this partition has a cost of $p \cdot \frac{B(B+1)}{2} - p(B-3)$.

For the other direction, assume there is a relaxed fair clustering $\mathcal{P}$ such that $\text{cost}(\mathcal{P}) \leq p \cdot \frac{B(B+1)}{2} - p(B-3)$. We prove that this clustering has to be not just relaxed fair but exactly fair. Note that $|V| = p(B+1)$ and $|E| = p(B-3)$. As the inter-cluster cost χ is non-negative, by Lemma 6.6 the intra-cluster cost has a lower bound of

$$\psi \geq \frac{(p(B+1))^2}{2|\mathcal{P}|} - \frac{p(B+1)}{2} - p(B-3).$$

As there are exactly p blue vertices and the relaxed fairness constraint requires putting at least one blue vertex in each cluster, we have $|\mathcal{P}| \leq p$. Hence,

$$\begin{aligned}\psi &\geq \frac{p(B+1)^2}{2} - \frac{p(B+1)}{2} - p(B-3)\\ &= p \cdot \frac{B(B+1)}{2} - p(B-3)\\ &\geq \text{cost}(\mathcal{P}).\end{aligned}$$

This implies that the inter-cluster cost of $\mathcal{P}$ is 0 and $|\mathcal{P}| = p$. Lemma 6.7 then gives that all clusters in $\mathcal{P}$ consist of exactly $B+1$ vertices. As each of the p

clusters has at least 1 blue vertex and there are p blue vertices in total, we know that each cluster consists of 1 blue and B red vertices. Since all trees are of size greater than $\frac{B}{4}$ and less than $\frac{B}{2}$, this implies each cluster consists of exactly one blue vertex and exactly three uncut trees with a total of B vertices. This way, such a clustering gives a solution to 3-Partition, so our instance is a yes-instance.

As the construction of the graph only takes polynomial time in the instance size, this implies our hardness result. ■

Indeed, we note that we obtain our hardness result for any fairness constraint that allows the exactly fair solution and enforces at least 1 vertex of each color in every cluster. The same holds when transferring our hardness proof for trees of diameter 4.

▶ **Theorem 6.9.** For every choice of $0 < p_1 \leq \frac{1}{c+1} \leq q_1 < 1$ and $0 < p_2 \leq \frac{c}{c+1} \leq q_2 < 1$, Relaxed Fair Correlation Clusteringon trees with diameter 4 and two colors in a ratio of $1 : c$ is NP-hard. ◀

Proof. We reduce from 3-Partition. We assume $B^2 > 16p$. We can do so as we obtain an equivalent instance of 3-Partition when multiplying all a_i and B by the same factor, here some value in $O(p)$. For every a_i we construct a star of a_i red vertices. Further, we let there be a star of p blue vertices. We obtain a tree of diameter 4 by connecting the center v of the blue star to all the centers of the red stars. Note that the ratio between blue and red vertices is $1 : B$. We now show that there is a relaxed fair clustering $\mathcal{P}$ such that

$$\text{cost}(\mathcal{P}) \leq \frac{pB^2 - pB}{2} + 7p - 7$$

if and only if the given instance is a yes-instance for 3-Partition.

If we have a yes-instance of 3-Partition, then there is a partition of the set of stars into p clusters of size B, each consisting of three stars. By assigning the blue vertices arbitrarily to one unique cluster each, we hence obtain an exact fair partition, which is thus also relaxed fair. We first compute the inter-cluster cost. We call an edge *blue* or *red* if it connects two blue or red vertices, respectively. We call an edge *blue-red* if it connects a blue and a red vertex. All $p - 1$ blue edges are cut. Further, all edges between v (the center of the blue star) and red vertices are cut except for the three stars to which v is assigned. This causes

$3p - 3$ more cuts, so the inter-cluster cost is

$$\chi = 4p - 4.$$

Each cluster consists of $B+1$ vertices and $B-3$ edges, except for the one containing v which has B edges. The intra-cluster cost is hence

$$\psi = p\left(\frac{B(B+1)}{2} - B + 3\right) - 3 = \frac{pB^2 - pB}{2} + 3p - 3.$$

Combining the intra- and inter-cluster costs yields the desired cost of

$$\begin{aligned}\operatorname{cost}(\mathcal{P}) &= \chi + \psi \\ &= \frac{pB^2 - pB}{2} + 7p - 7.\end{aligned}$$

For the other direction, assume there is a relaxed fair clustering $\mathcal{P}$ such that $\operatorname{cost}(\mathcal{P}) \leq \frac{pB^2-pB}{2} + 7p - 7$. We prove that this clustering has to be not just relaxed fair but exactly fair.

To this end, we first show $|\mathcal{P}| = p$. Because each cluster requires one of the p blue vertices, we have $|\mathcal{P}| \leq p$. Now, let χ denote the inter-cluster cost of $\mathcal{P}$. Note that $|V| = p(B+1)$ and $|E| = p(B-3) + 3p + p - 1 = p(B+1) - 1$. Then, by Lemma 6.6, we have

$$\begin{aligned}\psi &\geq \frac{(p(B+1))^2}{2|\mathcal{P}|} - \frac{p(B+1)}{2} - (p(B+1) - 1) + \chi \\ &= \frac{p^2B^2 + 2p^2B + p^2}{2|\mathcal{P}|} - \frac{3p(B+1)}{2} + 1 + \chi. \qquad (6.1)\end{aligned}$$

Note that the lower bound is decreasing in $|\mathcal{P}|$. If we had $|\mathcal{P}| \leq p - 1$, then

$$\psi \geq \frac{p^2B^2 + 2p^2B + p^2}{2(p-1)} - \frac{3p(B+1)}{2} + 1 + \chi.$$

As the inter-cluster cost χ is non-negative, we would thereby get

$$\operatorname{cost}(\mathcal{P}) \geq \frac{p^2B^2 + 2p^2B + p^2}{2(p-1)} - \frac{3p(B+1)}{2} + 1 + \chi$$

$$\geq \frac{p^2B^2 + 2p^2B + p^2}{2(p-1)} - \frac{3p^2B - 3pB + 3p^2 - 3p}{2(p-1)} + \frac{2p-2}{2(p-1)}$$
$$\geq \frac{p^2B^2 - p^2B - 2p^2 + 3pB + 5p - 2}{2(p-1)}.$$

However, we know

$$\begin{aligned}\text{cost}(\mathcal{P}) &\leq \frac{pB^2 - pB}{2} + 7p - 7 \\ &= \frac{p^2B^2 - pB^2 - p^2B + pB + 14p^2 - 14p - 14p + 14}{2(p-1)} \\ &= \frac{p^2B^2 - pB^2 - p^2B + pB + 14p^2 - 28p + 14}{2(p-1)}.\end{aligned}$$

Hence, $|\mathcal{P}| \leq p - 1$ holds only if

$$-2p^2 + 3pB + 5p - 2 \leq -pB^2 + pB + 14p^2 - 28p + 14$$

which is equivalent to

$$pB^2 - 16p^2 + 2pB + 33p - 16 \leq 0.$$

As we assume $B^2 > 16p$, this is always false, so $|\mathcal{P}| = p$.

Plugging this into Equation (6.1) yields

$$\begin{aligned}\psi &\geq \frac{pB^2 + 2pB + p}{2} - \frac{3p(B+1)}{2} + 1 + \chi \\ &= \frac{pB^2 - pB}{2} - p + 1 + \chi.\end{aligned}$$

As $\text{cost}(\mathcal{P}) = \chi + \psi$, we have

$$\frac{pB^2 - pB}{2} - p + 1 + 2\chi \leq \text{cost}(\mathcal{P}) \leq \frac{pB^2 - pB}{2} + 7p - 7, \tag{6.2}$$

which yields

$$\chi \leq 4p - 4.$$

As no two blue vertices are placed in the same cluster, the cuts between blue vertices incur an inter-cluster cost of exactly $p-1$. To estimate the number of cut blue-red edges, let a denote the number of red center vertices placed in the cluster of the blue center vertex v. Then, there are $3p-a$ of the $3p$ red edges cut. Let χ_r denote the number of cut red edges. Note that

$$\begin{aligned}\chi &= p-1+3p-a+\chi_r \\ &= 4p-a-1+\chi_r.\end{aligned}$$

We show that $a=3$. As $\chi \leq 4p-4$ we have $\chi_r - a \leq -3$. Hence, $a \geq 3$. Next, we bound χ_r by a. Let $\delta \in \mathbb{Z}$ be such that $B+\delta$ is the number of red vertices in the cluster containing the blue center vertex v. Then,

$$\begin{aligned}\chi_r &\geq \frac{aB}{4} - (B+\delta-a) \\ &= \frac{(a-4)B}{4} - \delta + a\end{aligned}$$

as each red center vertex is connected to at least $\frac{B}{4}$ red leaves but in the cluster of v there is only space for $B+\delta-a$ of them. First, assume $\delta \leq 0$. Then,

$$\chi_r - a \geq \frac{(a-4)B}{4}.$$

As we required $\chi_r - a \leq -3$, this gives $a < 4$. Now assume $\delta \geq 1$. Then, by Lemma 6.7, we have

$$\begin{aligned}\psi &\geq \frac{\delta^2|\mathcal{P}|}{2|\mathcal{P}|-2} + \frac{(p(B+1))^2}{2|\mathcal{P}|} - \frac{p(B+1)}{2} - m + \chi \\ &= \frac{\delta^2 p}{2p-2} + \frac{pB^2+2pB+p}{2} - \frac{3p(B+1)}{2} + \chi + 1\end{aligned}$$

as $p = |\mathcal{P}|$ and $m = n-1 = p(B+1)-1$. This yields

$$\frac{\delta^2 p}{2p-2} + \frac{pB^2-pB}{2} - p + 2\chi + 1 \leq \mathrm{cost}(\mathcal{P}) \leq \frac{pB^2-pB}{2} + 7p - 7.$$

so $\chi \leq 4p - 4 - \frac{\delta^2 p}{4p-4}$. Thereby,

$$\chi_r - a \leq -3 - \frac{\delta^2 p}{4p-4},$$

which implies

$$\frac{(a-4)B}{4} - \delta \leq -3 - \frac{\delta^2 p}{4p-4}$$

so

$$\frac{(a-4)B}{4} \leq -3 - \frac{\delta^2 p - \delta(4p-4)}{4p-4}$$

and we have

$$\frac{(a-4)B}{4} \leq -2 - \frac{p}{4p-4},$$

by noting that the right-hand side is decreasing in δ for $\delta \geq 1$ and plugging in $\delta = 1$. Hence, here also $a < 4$. Thus, we have proven $a = 3$, which also gives $\chi_r = 0$ and $\chi = 4p - 4$. So, not only do we have that $\text{cost}(\mathcal{P}) \leq \frac{pB^2-pB}{2} + 7p - 7$ but $\text{cost}(\mathcal{P}) = \frac{pB^2-pB}{2} + 7p - 7$. In Equation (6.2) we see that for $\chi = 4p - 4$ this hits exactly the lower bound established by Lemma 6.6. Hence, by Lemma 6.7, this implies that all clusters consist of exactly 1 blue and B red vertices and the clustering is exactly fair.

As $\chi_r = 0$, all red stars are complete. Given that every red star is of size at least $\frac{B}{4}$ and at most $\frac{B}{2}$, this means each cluster consists of exactly three complete red stars with a total number of B red vertices each and hence yields a solution to the 3-Partition instance. As the construction of the graph only takes polynomial time in the instance size and the constructed tree is of diameter 4, this implies our hardness result. ■

In the hardness proofs in this section, we argued that for the constructed instances clusterings that are relaxed fair, but not exactly fair would have a higher cost than exactly fair ones. However, this is not generally true. It does not even hold when limited to paths and two colors in a $1:1$ ratio, as illustrated in Figure 6.1. Because of this, we have little hope to provide a general scheme that

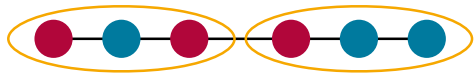

Figure 6.1: Exemplary path with a color ratio of $1:1$ where there is a $\frac{2}{3}$-relaxed fair clustering of cost 3 (marked by the orange lines) and the cheapest exactly fair clustering costs 4.

transforms all our hardness proofs from Chapter 4 to the relaxed fairness setting at once. Thus, we have to individually prove the hardness results in this setting as done for Theorems 6.8 and 6.9. We are optimistic that the other hardness results still hold in this setting, especially as the construction for Theorem 4.3 is similar to the ones employed in this section. We leave the task of transferring these results to future work.

6.3 Algorithm

We are also able to transfer the algorithmic result of Theorem 5.6 to a specific α-relaxed fairness setting. We exploit that the algorithm does not really depend on exact fairness but on the fact that there is an upper bound on the cluster size, which allows us to compute respective splittings. In the following, we show that such upper bounds also exist for α-relaxed fairness with two colors in a ratio of $1:1$ and adapt the algorithm accordingly. To compute the upper bound, we first prove Lemma 6.10, which analogously to Lemma 3.2 bounds the size of clusters but in uncolored forests. Using this lemma, with Lemma 6.11, we then prove an upper bound on the cluster size in minimum-cost α-relaxed fair clusterings for forests with two colors in ratio $1:1$.

▶ **Lemma 6.10.** Let $F = (V, E)$ be an n-vertex m-edge forest and let $\mathcal{P}_1 = \{V\}$. Further, let $S \subset V$ with $4 < |S| \leq n-3$ and let $\mathcal{P}_2 = \{S, V \setminus S\}$. Then, $\text{cost}(\mathcal{P}_1) > \text{cost}(\mathcal{P}_2)$. ◀

Proof. We have $\text{cost}(\mathcal{P}_1) = \frac{n(n-1)}{2} - m$ as there are $\frac{n(n-1)}{2}$ pairs of vertices and m edges, none of which is cut by $\mathcal{P}_1$. In the worst case, $\mathcal{P}_2$ cuts all of the at most $n-1$ edges in the forest. It has one cluster of size $|S|$ and one of size $n-|S|$, so

$$\text{cost}(\mathcal{P}_2) \leq n - 1 + \frac{(n-|S|)(n-|S|-1)}{2} + \frac{|S|(|S|-1)}{2} - (m-n-1)$$

$$= \frac{n(n-1)}{2} + \frac{-2n|S| + |S|^2 + |S|}{2} + \frac{|S|^2 - |S|}{2} - m + 2n - 2$$
$$= \frac{n(n-1)}{2} - n|S| + |S|^2 - m + 2n - 2.$$

Then, we have

$$\text{cost}(\mathcal{P}_1) - \text{cost}(\mathcal{P}_2) \geq n|S| - |S|^2 - 2n + 2 \geq (|S| - 2)n - |S|^2 + 2.$$

Note that the bound is increasing in n. As we have, $n \geq |S| + 3$, this gives

$$\begin{aligned}\text{cost}(\mathcal{P}_1) - \text{cost}(\mathcal{P}_2) &\geq (|S| - 2)(|S| + 3) - |S|^2 + 2 \\ &= |S| - 4 \\ &> 0.\end{aligned}$$

as we assume $|S| > 4$. This completes our proof. ■

With the knowledge of when it is cheaper to split a cluster, we now prove that also for α-relaxed Fair Correlation Clustering there is an upper bound on the cluster size in minimum-cost solutions in forests. The idea is to assume a cluster of a certain size and then argue that we can split it in a way that reduces the cost and keeps α-relaxed fairness.

▶ **Lemma 6.11.** Let F be a forest with two colors in a ratio of $1 : 1$. Let $0 < \alpha < 1$ and let $\hat{\alpha} \in \mathbb{N}$ be minimal such that $\frac{2\hat{\alpha}}{\alpha} \in \mathbb{N}$ and $\frac{2\hat{\alpha}}{\alpha} > 4$. Then, if $\mathcal{P}$ is a minimum-cost α-relaxed fair clustering on F, we have $|S| < 4\frac{\hat{\alpha}}{\alpha^2}$ for all $S \in \mathcal{P}$. ◀

Proof. Assume otherwise, i.e., there is a cluster S with $|S| \geq 4\frac{\hat{\alpha}}{\alpha^2}$. Let b and r denote the number of blue and red vertices in S, respectively, and assume w.l.o.g. that $b \leq r$. Because $|S| \geq 4\frac{\hat{\alpha}}{\alpha^2}$ we have $\frac{\alpha}{2} \geq \frac{2\hat{\alpha}}{\alpha|S|}$. Due to the α-relaxed fairness constraint, this yields $\frac{b}{|S|} \geq \frac{2\hat{\alpha}}{\alpha|S|}$ and thereby $r \geq b \geq \frac{2\hat{\alpha}}{\alpha}$.

Then, consider the clustering obtained by splitting off $\hat{\alpha}$ blue and $\frac{2\hat{\alpha}}{\alpha} - \hat{\alpha}$ red vertices of from S into a new cluster S_1 and let $S_2 = S \setminus S_1$. Note that we choose $\hat{\alpha}$ in a way that this is possible, i.e., that both sizes are natural numbers. As the cost induced by all edges with at most one endpoint in S remains the same and the cost induced by the edges with both endpoints in S decreases, as shown in Lemma 6.10, the new clustering is cheaper than $\mathcal{P}$. As we now prove that the new clustering is also α-relaxed Fair, this contradicts the optimality of $\mathcal{P}$.

We first prove the α-relaxed fairness of S_1. Regarding the blue vertices, we have a portion of $\frac{\hat{\alpha}}{\hat{\alpha}+\frac{2\hat{\alpha}}{\alpha}-\hat{\alpha}} = \frac{\alpha}{2}$ in S_1, which fits the α-relaxed fairness constraint. Regarding the red vertices, we have $\frac{\frac{2\hat{\alpha}}{\alpha}-\hat{\alpha}}{\hat{\alpha}+\frac{2\hat{\alpha}}{\alpha}-\hat{\alpha}} = 1-\frac{\alpha}{2}$, which fits the α-relaxed fairness constraint as $0 < \alpha < 1$, so $1-\frac{\alpha}{2} \geq \frac{\alpha}{2}$ and $1-\frac{\alpha}{2} = \frac{2\alpha-\alpha^2}{2\alpha} \leq \frac{1}{2\alpha}$.

Now we prove the α-relaxed fairness of S_2. The portion of blue vertices in S_2 is $\frac{b-\hat{\alpha}}{r+b-\frac{2\hat{\alpha}}{\alpha}}$, so we have to show that this value lays between $\frac{\alpha}{2}$ and $\frac{1}{2\alpha}$. We start with showing the value is at least $\frac{\alpha}{2}$ by proving $\frac{\alpha}{2} \cdot \left(r+b-\frac{2\hat{\alpha}}{\alpha}\right) \leq b-\hat{\alpha}$. As S is α-relaxed fair, we have $r \leq \frac{2b}{\alpha} - b$ because otherwise $\frac{b}{b+r} < \frac{b}{b+\frac{2b}{\alpha}-b} = \frac{\alpha}{2}$. Hence, we have

$$\frac{\alpha}{2} \cdot \left(r+b-\frac{2\hat{\alpha}}{\alpha}\right) \leq \frac{\alpha}{2} \cdot \left(\frac{2b}{\alpha} - b + b - \frac{2\hat{\alpha}}{\alpha}\right) = b-\hat{\alpha}.$$

Similarly, we show the ratio is at most $\frac{1}{2\alpha}$ by proving the equivalent statement of $2\alpha(b-\hat{\alpha}) \leq r+b-\frac{2\hat{\alpha}}{\alpha}$. As we assume $r \geq b$, we know

$$r+b-\frac{2\hat{\alpha}}{\alpha} \geq 2b-\frac{2\hat{\alpha}}{\alpha} \geq 2\left(b-\frac{\hat{\alpha}}{\alpha}-\left((1-\alpha)b+(\alpha^2-1)\frac{\hat{\alpha}}{\alpha}\right)\right) = 2\alpha(b-\hat{\alpha}).$$

The second step holds because we assume $b \geq \frac{2\hat{\alpha}}{\alpha} \geq \frac{\alpha\hat{\alpha}+\hat{\alpha}}{\alpha} = \frac{\frac{\hat{\alpha}}{\alpha}-\alpha\hat{\alpha}}{1-\alpha}$, so we have $(1-\alpha)b+(\alpha^2-1)\frac{\hat{\alpha}}{\alpha} \geq 0$. Now, we regard the portion of red vertices in S_2. It is $\frac{r-\left(\frac{2\hat{\alpha}}{\alpha}-\hat{\alpha}\right)}{r+b-\frac{2\hat{\alpha}}{\alpha}}$. We know

$$r \geq \frac{2\hat{\alpha}}{\alpha}$$

which is equivalent to

$$(1-\alpha)r \geq \frac{2\hat{\alpha}}{\alpha} - 2\hat{\alpha}$$

and thereby gives

$$r-\left(\frac{2\hat{\alpha}}{\alpha}+\hat{\alpha}\right) \geq \alpha r - \hat{\alpha}.$$

As $r \geq b$, this implies

$$r - \left(\frac{2\hat{\alpha}}{\alpha} + \hat{\alpha}\right) \geq \frac{\alpha}{2} \cdot \left(r + b - \frac{2\hat{\alpha}}{\alpha}\right),$$

so we have

$$\frac{r - \left(\frac{2\hat{\alpha}}{\alpha} - \hat{\alpha}\right)}{r + b - \frac{2\hat{\alpha}}{\alpha}} \geq \frac{\alpha}{2}.$$

It remains to prove that the ratio is at most $\frac{1}{2\alpha}$. We have

$$r \geq \frac{2\hat{\alpha}}{\alpha} - \hat{\alpha},$$

which is equivalent to

$$\left(2\alpha - 1 - \frac{\alpha}{2-\alpha}\right) r \leq 4\hat{\alpha} - 2\alpha\hat{\alpha} - \frac{2\hat{\alpha}}{\alpha}.$$

Note that $2\alpha - 1 - \frac{\alpha}{2-\alpha} = -\frac{2\alpha^2-4\alpha+2}{2-\alpha} = -\frac{2(\alpha-1)^2}{2-\alpha} < 0$. Further, note that $r \leq \frac{2b}{\alpha} - b$ gives $b \geq \frac{r}{\frac{2}{\alpha}-1} = \frac{\alpha r}{2-\alpha}$. With this, the above inequality implies

$$(2\alpha - 1)r - b \leq 4\hat{\alpha} - 2\alpha\hat{\alpha} - \frac{2\hat{\alpha}}{\alpha},$$

which is equivalent to

$$2\alpha \cdot \left(r - \left(\frac{2\hat{\alpha}}{\alpha} - \hat{\alpha}\right)\right) \leq r + b - \frac{2\hat{\alpha}}{\alpha}$$

and thereby yields

$$\frac{r - \left(\frac{2\hat{\alpha}}{\alpha} - \hat{\alpha}\right)}{r + b - \frac{2\hat{\alpha}}{\alpha}} \leq \frac{1}{2\alpha}.$$

With this, we have proven that both S_1 and S_2 are α-relaxed fair. As splitting S into

S_1 and S_2 remains α-relaxed fair and it of cheaper cost, this contradicts S being in any minimum-cost α-relaxed fair clustering and completes our proof. ■

We are now able to adapt the algorithm presented in Section 5.3 to solve Relaxed Fair Correlation Clusteringon forests with two colors in a ratio of $1:1$. While the original algorithm exploited that any optimum solution has fair clusters of minimum size, with Lemma 6.11 we are able to bound the clusters also in the α-relaxed setting.

Like the original algorithm, we first create a list of possible splittings. However, these splittings can contain not only components with one or two vertices, as we know would suffice for the exact fairness with two colors in a $1:1$ ratio, but each component may contain up to $4\frac{\hat{\alpha}}{\alpha^2}$ vertices with $\hat{\alpha}$ being the smallest natural number such that $\frac{2\hat{\alpha}}{\alpha} \in \mathbb{N}$ and $\frac{2\hat{\alpha}}{\alpha} > 4$ as defined in Lemma 6.11. In the following, we set $d = 4\frac{\hat{\alpha}}{\alpha^2}$ to refer to this maximum size of a cluster. In the second phase, it checks which of these splitting can be merged into an α-relaxed fair clustering and among these returns the one of minimum cost.

Splitting the forest. To get the optimal way to obtain a splitting of each possible coloring, we simply apply Lemma 5.8 and set $d_1 = d_2 = d$ as we know the optimum solution has to be among clusters with no more than d vertices of either color. This phase takes time in $\mathrm{O}\left(n^{2(d+1)^2+2d+2} \cdot \left((d+1)^2\right)^{2d}\right) = \mathrm{O}\left(n^{2d^2+6d+4} \cdot (d+1)^{4d}\right)$.

Assembling a fair clustering. In the second phase, we have to find a splitting in $D_r^{\emptyset}$ that can be transformed into an α-relaxed fair clustering and yields the minimum Correlation Clustering cost. As we tracked the minimum inter-cluster cost for each possible partition coloring of splittings in the first phase, we do not have to consider cutting more edges in this phase, because for the resulting splittings coloring we already have tracked a minimum inter-cluster cost. Hence, the only questions are whether a splitting is *assemblable*, i.e., whether its components can be merged such that it becomes an α-relaxed fair clustering, and, if so, what the cheapest way to do so is.

Regarding the first question, observe that the *assemblability* only depends on the partition coloring of the splitting. Hence, it does not hurt that in the first phase we tracked only all possible partition colorings of splittings and not all

possible splittings themselves. First, note that the coloring of a splitting may itself yield an α-relaxed fair clustering. We mark all such partition colorings as assemblable, taking time in $O\left(n^{d^2+1}\right)$. For the remaining partition colorings, we employ the following dynamic program.

Recall that the size of a partition coloring refers to the number of set colorings it contains (not necessarily the number of different set colorings). We decide assemblability for all possible partition colorings from smallest to largest. Note that each partition coloring is of size at least $\lceil \frac{n}{d} \rceil$. If it is of size exactly $\lceil \frac{n}{d} \rceil$, then there are no two set colorings that can be merged and still be of size at most d, as all other set colorings are of size at most d. Hence, in this case, a splitting is assemblable if and only if it is already an α-relaxed fair clustering so we have already marked the partition colorings correctly. Now, assume that we decided assemblability for all partition colorings of size $i \geq \lceil \frac{n}{d} \rceil$. We take an arbitrary partition coloring C of size $i+1$, which is not yet marked as assemblable. Then, it is assemblable if and only if at least two of its set colorings are merged together to form an α-relaxed fair clustering. In particular, it is assemblable if and only if there are two set colorings C_1, C_2 in C such that the coloring C' obtained by removing the set colorings C_1, C_2 from C and adding the set coloring of the combined coloring of C_1 and C_2 is assemblable. Note that C' is of size i. Given all assemblable partition colorings of size i, we therefore find all assemblable partition colorings of size $i+1$ by for each partition coloring of size i trying each possible way to split one of its set colorings into two. As there are at most i^{d^2} partition colorings of size i, this takes time in $O\left(i^{d^2} \cdot i \cdot 2^d\right)$. The whole dynamic program then takes time in $O\left(n^{d^2+1} \cdot 2^d\right) \subseteq O\left(n^{d^2+d+1}\right)$.

It remains to answer how we choose the assembling yielding the minimum cost. In the algorithm for exact fairness, we do not have to worry about that as there we could assume that the Correlation Clustering cost only depends on the inter-cluster cost. Here, this is not the case as the α-relaxed fairness allows clusters of varying size, so Lemma 3.1 does not apply. However, recall that we can write the Correlation Clustering cost of some partition $\mathcal{P}$ of the vertices as $\sum_{S \in \mathcal{P}} \frac{|S|(|S|-1)}{2} + 2\chi$, where χ is the inter-cluster cost. The cost hence only depends on the inter-cluster cost and the sizes of the clusters, which in turn depends on the partition coloring. To compute the cost of a splitting, we take the inter-cluster cost computed in the first phase for χ. Once more, we neglect decreasing inter-cluster cost due to the merging of clusters as the

resulting splitting is also considered in the array produced in the first phase. By an argument based on the Cauchy-Schwarz Inequality, we see that merging clusters only increases the value of $\sum_{S \in \mathcal{P}} \frac{|S|(|S|-1|)}{2}$ as we have fewer but larger squares. Hence, the cheapest cost obtainable from a splitting which is itself α-relaxed fair is just this very clustering. If a splitting is assemblable but not α-relaxed fair itself, the sum is the minimum among all the values of the sums of α-relaxed fair splittings it can be merged into. This value is easily computed by not only passing down assemblability but also the value of this sum in the dynamic program described above and taking the minimum if there are multiple options for a splitting. This does not change the running time asymptotically and the running time of the second phase is dominated by the one of the first phase.

The complete algorithm hence runs in time in $O\left(n^{2d^2+6d+4} \cdot (d+1)^{4d}\right)$.

▶ **Theorem 6.12.** Let F be an n-vertex forest in which the vertices are colored with two colors in a ratio of $1:1$. Then α-relaxed Fair Correlation Clustering on F can be solved in time in $O\left(n^{2d^2+6d+4} \cdot (d+1)^{4d}\right)$, where $d = 4\frac{\hat{\alpha}}{\alpha^2}$ and $\hat{\alpha} \in \mathbb{N}$ is minimal such that $\frac{2\hat{\alpha}}{\alpha} \in \mathbb{N}$ and $\frac{2\hat{\alpha}}{\alpha} > 4$. ◀

We are confident that Lemma 6.11 can be generalized such that for an arbitrary number of colors in arbitrary ratios the maximum cluster size is bounded by some function in α and the color ratio. Given the complexity of this lemma for the $1:1$ case, we leave this task open to future work. If such a bound is proven, then the algorithmic approach employed in Theorem 6.12 is applicable to arbitrarily colored forests. Similarly, bounds on the cluster size in the more general relaxed fair clusterings can be proven. As an intermediate solution, we note that for Relaxed Fair Correlation Clusteringwe can employ the approach used for α-relaxed Fair Correlation Clustering by setting α large enough to contain all allowed solutions and filtering out solutions that do not match the relaxed fairness constraint in the assembling phase. We do not give this procedure explicitly in this thesis as we suspect for these cases it is more promising to calculate the precise upper bound on the maximum cluster size and perform the algorithm accordingly instead of reducing to the α-relaxed variant.

7 Approximating Fair Correlation Clusterings

So far, we have concentrated on finding an optimal solution to Fair Correlation Clustering in various instances. Approximation algorithms that do not necessarily find an optimum but near-optimum solutions efficiently are often used as a remedy for hard problems, for example, the 2.06-approximation to (unfair) Correlation Clustering [Cha+15]. In this chapter, we find that just taking any fair clustering is a quite close approximation and the approximation becomes even closer to the optimum if the minimum size of any fair cluster, as given by the color ratio, increases.

Formally, a problem is an optimization problem if for every instance I there is a set of permissible solutions $S(I)$ and an objective function $m\colon S(I) \to \mathbb{R}_{>0}$ assigning a score to each solution. Then, some $S \in S(I)$ is an optimal solution if it has the highest or lowest score among all permissible solutions, depending on the problem definition. We call the score of this solution $m^*(I)$. For example, for Fair Correlation Clustering, the instance is given by a graph with colored vertices, every fair clustering of the vertices is a permissible solution, the score is the Correlation Clustering cost, and the objective is to minimize this cost [2]. An α-approximation an optimization problem is an algorithm that, for each instance I, outputs a permissible solution $S \in S(I)$ such that $\frac{1}{\alpha} \leq \frac{m(S)}{m^*(I)} \leq \alpha$. For Fair Correlation Clustering in particular, this means the algorithm outputs a fair clustering with a cost of at most α times the minimum clustering cost.

APX (abbreviation of *approximable*) is the class of problems that admit an α-approximation with $\alpha \in \mathrm{O}(1)$. A PTAS, a polynomial-time approximation scheme, is an algorithm that for each optimization problem instance as well as parameter $\varepsilon > 0$ computes a $(1+\varepsilon)$-approximation for a minimization problem or a $(1-\varepsilon)$-approximation for a maximization problem in time in $\mathrm{O}\big(n^{f(\varepsilon)}\big)$, for some computable function f depending only on ε. We use the name PTAS also to refer to the class of optimization problems admitting a PTAS. An optimization

2 We note that the cost is possibly 0, which contradicts the definition $m\colon S(I) \to \mathbb{R}_{>0}$. However, every 0-cost clustering simply consists of all maximal-sized connected components in the graph. In this chapter, we exclude these trivial instances.

problem L is called APX-hard if every problem in APX has a PTAS-reduction to L, i.e., a PTAS for L implies there is a PTAS for every problem in APX. If L is additionally in APX itself, L is called APX-complete. By definition, we have PTAS $\subseteq$ APX. Further, PTAS $\neq$ APX unless P = NP.

We find that taking any fair clustering of a forest yields a good approximation.

▶ **Theorem 7.1.** Let F be an n-vertex m-edge forest with $k \geq 2$ colors in a ratio of $c_1 : c_2 : \ldots : c_k$ and $d = \sum_{i=1}^{k} c_i \geq 4$. Then, there is a $\frac{(d^2-d)n+2dm}{(d^2-5d+4)n+2dm}$-approximation for Fair Correlation Clustering on F computable in time in $O(n)$. ◀

Proof. By first sorting the vertices by color and then iteratively adding the next c_i vertices of each color i to the next cluster, we obtain a fair clustering $\mathcal{P}$ with clusters of size d in linear time. In the worst-case, $\mathcal{P}$ cuts all m edges. Hence, by Lemma 3.1, we have

$$\begin{aligned} \text{cost}(\mathcal{P}) &\leq \frac{(d-1)n}{2} - m + 2m \\ &= \frac{(d-1)n}{2} + m. \end{aligned}$$

We compare this cost to the one of a minimum-cost fair clustering $\mathcal{P}^*$. By Lemma 3.2, $\mathcal{P}^*$ to consist of clusters of size d. Each of the $\frac{n}{d}$ clusters contains at most $d-1$ edges due to the forest structure. Hence, at most $\frac{n}{d} \cdot (d-1)$ edges are placed inside a cluster. Then, for the inter-cluster cost, we have

$$\chi \geq m - \frac{n}{d} \cdot (d-1) = \frac{n}{d} - n + m.$$

Then, Lemma 3.1 gives

$$\begin{aligned} \text{cost}(\mathcal{P}^*) &\geq \frac{(d-1)n}{2} - m + 2\left(\frac{n}{d} - n + m\right) \\ &= \frac{(d-5)n}{2} + \frac{2n}{d} + m. \end{aligned}$$

Thereby, $\mathcal{P}$ yields an α-approximation to Fair Correlation Clustering, where

$$\alpha = \left(\frac{(d-1)n}{2} + m\right) \Big/ \left(\frac{(d-5)n}{2} + \frac{2n}{d} + m\right)$$

$$= \left(\frac{(d^2-d)n+2dm}{2d}\right) / \left(\frac{(d^2-5d+4)n+2dm}{2d}\right)$$
$$= \frac{(d^2-d)n+2dm}{(d^2-5d+4)n+2dm}.$$

■

Observe that α is decreasing in d for $d \geq 4$ and converges to 1 as $d \to \infty$. Further, for $d = 5$ we obtain $\alpha = \frac{20n+10m}{4n+10m} < 5$. Thus, for $d \geq 5$ we have a 5-approximation to Fair Correlation Clustering on forests. For $d = 4$, α becomes linear in $\frac{m}{n}$ and for smaller d it is not necessarily positive or not even defined if $(d^2-5d+4)n + 2dm = 0$. This is because if there are very small clusters, then in forests there are solutions of almost no cost. If $d = 2$, i.e., there are two colors in a $1:1$ ratio, there are even forests with a cost of 0, namely the ones where all vertices have degree 1 and each edge connects 2 vertices of different colors. A solution cutting every edge is then much worse than an optimum solution. If the factor becomes negative or not defined, this is due to us bounding the inter-cluster cost of the optimum clustering by $\frac{n}{d} - n + m$, which is possibly negative, while the inter-cluster cost is guaranteed to be non-negative.

On trees, however, if the clusters are small even an optimum solution has to cut some edges as now there always are edges between the clusters. Hence, in this case, we obtain a good approximation for all possible d. Note that the proof of Theorem 7.1 does not really require $d \geq 4$ but for $d < 4$ the approximation factor is just not helpful or defined. This changes, if we assume the forest to be a tree and plug in $m = n - 1$.

▶ **Corollary 7.2.** Let T be an n-vertex tree with $k \geq 2$ colors in a ratio of $c_1 : c_2 : \ldots : c_k$ and $d = \sum_{i=1}^{k} c_i$. Then, there is a $\frac{(d^2+d)n-2d}{(d^2-3d+4)n-2d}$-approximation to Fair Correlation Clustering on T that is computed in time in $O(n)$. ◀

Now, the approximation factor is still decreasing in d and converges to 1 as $d \to \infty$. However, it is positive and defined for all $d \geq 2$. For $d = 2$ we obtain $\frac{6n-4}{2n-4} < 3$. Therefore, we have a 3-approximation to Fair Correlation Clustering on trees.

Nevertheless, also our results for forest suffice to place Fair Correlation Clustering in APX and even PTAS. First, for $d \geq 5$ we have a 5-approximation to Fair Correlation Clustering on forests. If $d \leq 4$, a minimum-cost fair clustering is found on the forest in polynomial time by Theorem 5.6. Hence,

Fair Correlation Clustering on forests is in APX. Next, recall that the larger the minimum fair cluster size d, the better the approximation becomes. Recall that our dynamic program for Theorem 5.6 has better running time the smaller the value d. By combining these results, we obtain a PTAS for Fair Correlation Clustering on forests. This contrasts Fair Correlation Clustering on general graphs, as even unfair Correlation Clustering is APX-hard there [CGW05] and therefore does not admit a PTAS unless P = NP.

▶ **Theorem 7.3.** There is a PTAS for Fair Correlation Clustering on forests. ◀

Proof. If $d \leq 4$, we find a minimum-cost fair clustering in polynomial time by Theorem 5.6. Else, if $\frac{(d^2-d)n+2dm}{(d^2-5d+4)n+2dm} \leq 1+\varepsilon$, it suffices to return any fair clustering by Theorem 7.1. Otherwise, we have $d \geq 5$ and

$$\begin{aligned} 1+\varepsilon &< \frac{(d^2-d)n+2dm}{(d^2-5d+4)n+2dm} \\ &< \frac{(d^2-d)n}{(d^2-5d)n} \\ &= \frac{d-1}{d-5}. \end{aligned}$$

Then,

$$d-5+d\varepsilon-5\varepsilon < d-1,$$

so

$$d\varepsilon < 5\varepsilon + 4$$

and thus

$$d < \frac{5}{\varepsilon} + 4.$$

Hence, by Theorem 5.6, we find a minimum-cost fair clustering in time in $O\left(n^{f(5\varepsilon^{-1}+4)}\right)$ for some computable function f independent from n.

In all cases, we find a fair clustering with a cost of at most $1+\varepsilon$ times the

minimum Correlation Clustering cost and take at most time in $O\left(n^{f(4\varepsilon^{-1}+5)}\right)$, so our procedure is a PTAS. ■

8 Conclusions & Outlook

In this thesis, we found that Fair Correlation Clustering on trees and thereby forests is NP-hard. It remains NP-hard for trees of constant degree or constant diameter, and, for certain color distributions, it is also NP-hard on paths.

On the other hand, when parameterized by the minimum size d of any fair cluster as implied by the color ratio, we give an algorithm that places Fair Correlation Clustering on forests in XP, i.e., has a polynomial running time in the input size if d is constant. Further, we provide an XP-algorithm for Fair Correlation Clustering parameterized by the number of clusters if the color ratio is $1 : c$ for some $c \in \mathbb{N}_{>0}$. Hence, if d is very large, i.e., $d \in \Theta(n)$, there is only a constant number of clusters and this algorithm has a polynomial running time in the graph size.

For our main algorithms and hardness results, we proved that they essentially still hold when the fairness constraint is relaxed, so the hardness is not due to the strict fairness definition.

Lastly, we showed that the cost of any fair clustering on forests does not differ too much from an optimal fair clustering, and thereby yields a good approximation. As this approximation improves as the minimum size d of any fair cluster increases, this gives a PTAS when combined with our XP-algorithm.

Our hardness results even for certain types of trees and forests justify the use of approximation algorithms for Fair Correlation Clustering, even when solving very restricted instances. Ultimately, we hope that the insights gained from these proofs as well as our proposed algorithms prove helpful to the future development of approximation, parameterized, and randomized algorithms to solve Fair Correlation Clustering on more general graphs.

As a first step to further generalize our results, with Lemma 3.4, we generalize our key insight, that there is a minimum-cost fair clustering with clusters of bounded size from forests to bipartite graphs. We wonder whether this proves helpful in developing algorithms for bipartite graphs with other color ratios than $1 : 1$ as our tree dynamic program approach is not easily transferable to bipartite graphs.

Parameterized algorithms are another approach to solving more general instances. While our dynamic programs can be regarded as parameterized by the color ratio, other parameters might yield interesting results. There are FPT-algorithms for Cluster Editing parameterized by the solution cost [BB13]. Possibly, future research might provide similar results for Fair Correlation Clustering, allowing to efficiently solve instances that are quite close to a perfect clustering. Further, we thought about parameterizing by treewidth and giving a treewidth dynamic program with a similar strategy to our tree dynamic programs. However, it is questionable whether such an approach would work, as it might be difficult or impossible to bound the cluster size. Such a solution would also be surprising since, to the best of our knowledge, so far even for normal, unfair Correlation Clustering[3] and for the related Max Dense Graph Partition [DBM12] no treewidth approaches have been proposed.

Besides, this thesis opens various other questions to be answered. While we give two XP-algorithms for Fair Correlation Clustering on forests, we are wondering whether Fair Correlation Clustering for these parameters can also be placed in FPT, i.e., whether there are algorithms with running time $f(p) \cdot n^{O(1)}$ for some function f independent from n, where p is either the minimum size of any fair cluster or the number of clusters.

On a side note, it is interesting how Fair Correlation Clustering behaves on paths. While we obtained NP-hardness for a particular color distribution from the related Paint Shop Problem For Words (PPW), the question of whether Fair Correlation Clustering on paths with for example two colors in a ratio of $1 : c$ is efficiently solvable or not is, to the best of our knowledge, still open. However, we suppose that this question is rather answered in the study of PPW or the related Necklace Splitting than by research on Fair Correlation Clustering. This would close one of the two remaining open cells in our result tables. If Fair Correlation Clustering with a color ratio of $1 : c$ can indeed be efficiently solved on paths, it would be interesting to see whether it becomes hard for maximum degree 3 or 4. The other open cell from our tables concerns general graphs with a color ratio of $1 : 2$. As the $1 : 1$ case is already NP-hard, the same holds most likely for $1 : 2$. We do not give a hardness proof as it would

3 No algorithm for complete Correlation Clustering has been proposed. Xin [Xin11] gives a treewidth algorithm for incomplete Correlation Clustering for the treewidth of the graph of all positively and negatively labeled edges.

probably look very similar to the one provided by Ahmadi et al. [Ahm+20a] for the $1:1$ case and not give any additional insight.

Further, some algorithms and hardness results remain to be transferred to the relaxed fairness setting. Regarding hardness, this concerns trees of degree 5 (Theorem 4.3) and general graphs (Theorems 4.5 and 4.6). Regarding algorithms, we show how the case of two colors in a ratio of $1:1$ is transferred to α-relaxed Fair Correlation Clustering on forests. More generally, for k colors in a ratio of $c_1 : c_2 : \ldots : c_k$, we are optimistic that the maximum size of each cluster in a minimum-cost fair clustering on forests is bounded by a function in α and $d = \sum_{i=1}^{k} c_i$. This would yield a dynamic program just like we have shown for the $1:1$ case. Analogously, the algorithmic results are to be generalized from α-relaxed Fair Correlation Clustering to Relaxed Fair Correlation Clustering. Also, our results on trees of diameters 2 and 3 (Theorem 5.2) are likely to still hold in the relaxed fairness setting. As the clusters are not of fixed size, however, such a proof seems not to be trivial.

Bibliography

[ACN08] Nir Ailon, Moses Charikar, and Alantha Newman. **Aggregating inconsistent information: Ranking and clustering**. *Journal of the ACM* 55:5 (Nov. 2008), 23:1–23:27 (see page 6).

[Ahm+20a] Saba Ahmadi, Sainyam Galhotra, Barna Saha, and Roy Schwartz. **Fair Correlation Clustering**. arXiv preprint. *arXiv:2002.03508* (Feb. 2020) (see pages 5, 11, 35, 36, 63, 89).

[Ahm+20b] Sara Ahmadian, Alessandro Epasto, Ravi Kumar, and Mohammad Mahdian. **Fair Correlation Clustering**. In: *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*. June 2020, 4195–4205 (see pages 10, 11, 17).

[Alo87] Noga Alon. **Splitting necklaces**. *Advances in Mathematics* 63:3 (Mar. 1987), 247–253 (see page 33).

[Ami04] Noga Amit. **The bicluster graph editing problem**. Master's thesis. Tel Aviv University, Sept. 2004 (see page 7).

[AW86] Noga Alon and Douglas B. West. **The Borsuk-Ulam Theorem and Bisection of Necklaces**. *Proceedings of the American Mathematical Society* 98:4 (Dec. 1986), 623–628 (see page 33).

[Bac+13] Yoram Bachrach, Pushmeet Kohli, Vladimir Kolmogorov, and Morteza Zadimoghaddam. **Optimal Coalition Structures in Cooperative Graph Games**. In: *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*. July 2013, 81–87 (see page 8).

[Bac+19] Arturs Backurs, Piotr Indyk, Krzysztof Onak, Baruch Schieber, Ali Vakilian, and Tal Wagner. **Scalable Fair Clustering**. In: *Proceedings of the 36th International Conference on Machine Learning*. May 2019, 405–413 (see page 10).

[Bas+16] Lucas Bastos, Luiz Satoru Ochi, Fábio Protti, Anand Subramanian, Ivan César Martins, and Rian Gabriel S. Pinheiro. **Efficient algorithms for cluster editing**. *Journal of Combinatorial Optimization* 31:1 (Jan. 2016), 347–371 (see pages 7, 35).

[BB13] Sebastian Böcker and Jan Baumbach. **Cluster Editing**. In: *The Nature of Computation. Logic, Algorithms, Applications*. July 2013, 33–44 (see page 88).

[BBC04] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. **Correlation Clustering**. *Machine Learning* 56:1–3 (July 2004), 89–113 (see pages 6, 7).

[BEH06] Paul Simon Bonsma, Thomas Epping, and Winfried Hochstättler. **Complexity results on restricted instances of a paint shop problem for words**. *Discrete Applied Mathematics*. 2nd Cologne/Twente Workshop on Graphs and Combinatorial Optimization (CTW 2003) 154:9 (June 2006), 1335–1343 (see page 34).

[Ber+18] Ioana O. Bercea, Martin Groß, Samir Khuller, Aounon Kumar, Clemens Rösner, Daniel R. Schmidt, and Melanie Schmidt. **On the cost of essentially fair clusterings**. arXiv preprint. *arXiv:1811.10319* (Nov. 2018) (see page 10).

[Ber+19] Suman K. Bera, Deeparnab Chakrabarty, Nicolas J. Flores, and Maryam Negahbani. **Fair algorithms for clustering**. In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Dec. 2019, 4954–4965 (see pages 10, 63).

[BGG22] Francesco Bonchi, David García-Soriano, and Francesco Gullo. **Correlation Clustering**. Morgan & Claypool Publishers, Mar. 2022 (see pages 6, 16).

[BGW17] André Berger, Alexander Grigoriev, and Andrej Winokurow. **A PTAS for the Cluster Editing Problem on Planar Graphs**. In: *International Workshop on Approximation and Online Algorithms*. Jan. 2017, 27–39 (see page 8).

[BJ13] Jeremias Berg and Matti Järvisalo. **Optimal Correlation Clustering via MaxSAT**. In: *2013 IEEE 13th International Conference on Data Mining Workshops*. Dec. 2013, 750–757 (see page 8).

[Böc+09] Sebastian Böcker, Sebastian Briesemeister, Quang Bao Anh Bui, and Anke Truss. **Going weighted: Parameterized algorithms for cluster editing**. *Theoretical Computer Science*. Combinatorial Optimization and Applications 410:52 (Dec. 2009), 5467–5480 (see page 8).

[Bon+15] Francesco Bonchi, Aristides Gionis, Francesco Gullo, Charalampos E. Tsourakakis, and Antti Ukkonen. **Chromatic Correlation Clustering**. *ACM Transactions on Knowledge Discovery from Data* 9:4 (June 2015), 34:1–34:24 (see page 8).

[BST15] Max Bannach, Christoph Stockhusen, and Till Tantau. **Fast Parallel Fixed-parameter Algorithms via Color Coding**. In: *10th International Symposium on Parameterized and Exact Computation*. Sept. 2015, 224–235 (see page 8).

[BSY99] Amir Ben-Dor, Ron Shamir, and Zohar Yakhini. **Clustering Gene Expression Patterns**. *Journal of Computational Biology* 6:3–4 (Oct. 1999), 281–297 (see page 6).

[CGW05] Moses Charikar, Venkatesan Guruswami, and Anthony Wirth. **Clustering with qualitative information**. *Journal of Computer and System Sciences*. Learning Theory 2003 71:3 (Oct. 2005), 360–383 (see pages 6, 84).

[Cha+15] Shuchi Chawla, Konstantin Makarychev, Tselil Schramm, and Grigory Yaroslavtsev. **Near Optimal LP Rounding Algorithm for CorrelationClustering on Complete and Complete k-partite Graphs**. In: *Proceedings of the forty-seventh annual ACM symposium on Theory of Computing*. June 2015, 219–228 (see pages 7, 81).

[Chi+17] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, and Sergei Vassilvitskii. **Fair clustering through fairlets**. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Dec. 2017, 5036–5044 (see pages 10, 11).

[CM12] Jianer Chen and Jie Meng. **A 2k kernel for the cluster editing problem**. *Journal of Computer and System Sciences*. JCSS Knowledge Representation and Reasoning 78:1 (Jan. 2012), 211–220 (see page 8).

[Cyg+19] Marek Cygan, Marcin Mucha, Karol Węgrzycki, and Michał Włodarczyk. **On Problems Equivalent to (min,+)-Convolution**. *ACM Transactions on Algorithms* 15:1 (Jan. 2019), 14:1–14:25 (see page 48).

[DBM12] Julien Darlay, Nadia Brauner, and Julien Moncel. **Dense and sparse graph partition**. *Discrete Applied Mathematics* 160:16 (Nov. 2012), 2389–2396 (see page 88).

[Din+22] Michael Dinitz, Aravind Srinivasan, Leonidas Tsepenekas, and Anil Vullikanti. **Fair Disaster Containment via Graph-Cut Problems**. In: *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*. May 2022, 6321–6333 (see page 10).

[EHO04] Thomas Epping, Winfried Hochstättler, and Peter Oertel. **Complexity results on a paint shop problem**. *Discrete Applied Mathematics* 136:2-3 (Feb. 2004), 217–226 (see pages 34, 35).

[Esm+20] Seyed A. Esmaeili, Brian Brubach, Leonidas Tsepenekas, and John P. Dickerson. **Probabilistic fair clustering**. In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. Dec. 2020, 12743–12755 (see page 10).

[Fel+15] Michael Feldman, Sorelle A. Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. **Certifying and Removing Disparate Impact**. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Aug. 2015, 259–268 (see page 10).

[FF15] Andreas Emil Feldmann and Luca Foschini. **Balanced Partitions of Trees and Applications**. *Algorithmica* 71:2 (Feb. 2015), 354–376 (see pages 9, 31, 33).

[FKN22] Vincent Froese, Leon Kellerhals, and Rolf Niedermeier. **Modification-Fair Cluster Editing**. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. June 2022, 6631–6638 (see page 9).

[FM21] Zachary Friggstad and Ramin Mousavi. **Fair Correlation Clustering with Global and Local Guarantees**. In: *Algorithms and Data Structures*. Aug. 2021, 414–427 (see page 12).

[For10] Santo Fortunato. **Community detection in graphs**. *Physics Reports* 486:3 (Feb. 2010), 75–174 (see page 6).

[GG06] Ioannis Giotis and Venkatesan Guruswami. **Correlation Clustering with a Fixed Number of Clusters**. *Theory of Computing* 2:1 (Oct. 2006), 249–266 (see page 9).

[GJ79] Michael R. Garey and David S. Johnson. **Computers and intractability: A guide to the theory of NP-completeness**. W. H. Freeman, Jan. 1979 (see pages 27, 37).

[GSV22] Mehrdad Ghadiri, Mohit Singh, and Santosh S. Vempala. **Constant-Factor Approximation Algorithms for Socially Fair *k*-Clustering**. arXiv preprint. *arXiv:2206.11210* (June 2022) (see page 10).

[HK71] John E. Hopcroft and Richard M. Karp. **A nˆ5/2 algorithm for maximum matchings in bipartite**. In: *12th Annual Symposium on Switching and Automata Theory*. Oct. 1971, 122–125 (see page 42).

[HL22] Tesshu Hanaka and Michael Lampis. **Hedonic Games and Treewidth Revisited**. In: *30th Annual European Symposium on Algorithms*. Aug. 2022, 64:1–64:16 (see page 8).

[KAM19] Matthäus Kleindessner, Pranjal Awasthi, and Jamie Morgenstern. **Fair k-Center Clustering for Data Summarization**. In: *Proceedings of the 36th International Conference on Machine Learning*. May 2019, 3448–3457 (see page 10).

[Kel+21] Leon Kellerhals, Tomohiro Koana, André Nichterlein, and Philipp Zschoche. **The PACE 2021 Parameterized Algorithms and Computational Experiments Challenge: Cluster Editing**. In: *16th International Symposium on Parameterized and Exact Computation*. Sept. 2021, 26:1–26:18 (see page 8).

[Klo+21] Nicolas Klodt, Lars Seifert, Arthur Zahn, Katrin Casel, Davis Issac, and Tobias Friedrich. **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*. Aug. 2021, 882–891 (see page 9).

[KU12] Christian Komusiewicz and Johannes Uhlmann. **Cluster editing with locally bounded modifications**. *Discrete Applied Mathematics* 160:15 (Oct. 2012), 2259–2270 (see page 7).

[Man10] Bassel Mannaa. **Cluster editing problem for points on the real line: A polynomial time algorithm**. *Information Processing Letters* 110:21 (Oct. 2010), 961–965 (see page 7).

[PS22] Dana Pessach and Erez Shmueli. **A Review on Fairness in Machine Learning**. *ACM Computing Surveys* 55:3 (Feb. 2022), 51:1–51:44 (see page 10).

[Rég83] Simon Régnier. **Sur quelques aspects mathématiques des problèmes de classification automatique**. *Mathématiques et Sciences Humaines* 82 (Jan. 1983), 31–44 (see page 6).

[Sch07] Satu Elisa Schaeffer. **Graph clustering**. *Computer Science Review* 1:1 (Aug. 2007), 27–64 (see page 6).

[SSS20] Melanie Schmidt, Chris Schwiegelshohn, and Christian Sohler. **Fair Coresets and Streaming Algorithms for Fair k-means**. In: *International Workshop on Approximation and Online Algorithms*. Jan. 2020, 232–251 (see page 10).

[SST04] Ron Shamir, Roded Sharan, and Dekel Tsur. **Cluster graph modification problems**. *Discrete Applied Mathematics*. Discrete Mathematics and Data Mining 144:1 (Nov. 2004), 173–182 (see page 9).

[SZ22] Roy Schwartz and Roded Zats. **Fair Correlation Clustering in General Graphs**. In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Sept. 2022, 37:1–37:19 (see page 12).

[Vai21] Michael Vaichenker. **Hardness and Approximation Results for Fair Ratio-Cut**. Master's thesis. Hasso-Plattner-Institute, University of Potsdam, Nov. 2021 (see page 10).

[VWG20] Nate Veldt, Anthony Wirth, and David F. Gleich. **Parameterized Correlation Clustering in Hypergraphs and Bipartite Graphs**. In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Aug. 2020, 1868–1876 (see page 7).

[Xin11] Xiao Xin. **An FPT Algorithm for the Correlation Clustering Problem**. *Key Engineering Materials* 474–476 (Apr. 2011), 924–927 (see pages 8, 88).

[XT15] Dongkuan Xu and Yingjie Tian. **A Comprehensive Survey of Clustering Algorithms**. *Annals of Data Science* 2:2 (June 2015), 165–193 (see page 6).

[Zah64] Charles T. Zahn Jr. **Approximating Symmetric Relations by Equivalence Relations**. *Journal of the Society for Industrial and Applied Mathematics* 12:4 (Dec. 1964), 840–847 (see page 6).

[Zik+21] Imtiaz Masud Ziko, Jing Yuan, Eric Granger, and Ismail Ben Ayed. **Variational Fair Clustering**. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. May 2021, 11202–11209 (see page 10).

Declaration of Authorship

I hereby declare that this thesis is my own unaided work. All direct or indirect sources used are acknowledged as references.

Potsdam, September 27, 2022 _____

Simon Wietheger