ABSTRACT

Title of Dissertation: SOCIAL ASPECTS OF ALGORITHMS: FAIRNESS, DIVERSITY, AND RESILIENCE TO STRATEGIC BEHAVIOR

Saba Ahmadi
Doctor of Philosophy, 2021

Dissertation Directed by: Professor Samir Khuller
Department of Computer Science

With algorithms becoming ubiquitous in our society, it is important to ensure that they are compatible with our social values. In this thesis, we study some of the social aspects of algorithms including fairness, diversity, and resilience to strategic behavior of individuals.

Lack of diversity has a potential impact on discrimination against marginalized groups. Inspired by this issue, in the first part of this thesis, we study a notion of diversity in bipartite matching problems. Bipartite matching where agents on one side of a market are matched to one or more agents or items on the other side, is a classical model that is used in myriad application areas such as healthcare, advertising, education, and general resource allocation. In particular, we consider an application of matchings where a firm wants to hire,

i.e. match, some workers for a number of teams. Each team has a demand that needs to be satisfied, and each worker has multiple features (e.g., country of origin, gender). We ask the question of how to assign workers to the teams in an efficient way, i.e. low-cost matching, while forming diverse teams with respect to all the features. Inspired by previous work, we balance whole-match diversity and economic efficiency by optimizing a supermodular function over the matching. Particularly, we show when the number of features is given as part of the input, this problem is NP-hard, and design a pseudo-polynomial time algorithm to solve this problem.

Next, we focus on studying fairness in optimization problems. Particularly, in this thesis, we study two notions of fairness in an optimization problem called *correlation clustering*. In *correlation clustering*, given an edge-weighted graph, each edge in addition to a weight has a positive or negative label. The goal is to obtain a clustering of the vertices into an arbitrary number of clusters that minimizes disagreements which is defined as the total weight of negative edges trapped inside a cluster plus the sum of weights of positive edges between different clusters. In the first fairness notion, assuming each node has a color, i.e. feature, our aim is to generate clusters with minimum disagreements, where the distribution of colors in each cluster is the same as the global distribution. Next, we switch our attention to a min-max notion

of fairness in *correlation clustering*. In this notion of fairness, we consider a cluster-wise objective function that asks to minimize the maximum number of disagreements of each cluster. In this notion, the goal is to respect the quality of each cluster. We focus on designing approximation algorithms for both of these notions.

In the last part of this thesis, we take into consideration, the vulnerability of algorithms to manipulation and gaming. We study the problem of how to learn a linear classifier in presence of strategic agents that desire to be classified as positive and that are able to modify their position by a limited amount, making the classifier not be able to observe the true position of agents but rather a position where the agent pretends to be. We focus on designing algorithms with a bounded number of mistakes for a few different variations of this problem.

SOCIAL ASPECTS OF ALGORITHMS: FAIRNESS, DIVERSITY, AND RESILIENCE TO STRATEGIC BEHAVIOR

by

Saba Ahmadi

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2021

Advisory Committee:
Professor Samir Khuller, Chair/Advisor
Professor Avrim Blum
Professor John P. Dickerson
Professor David Mount
Professor Louiqa Raschid

Acknowledgments

First and foremost, I would like to thank my advisor Samir Khuller for his constant and reliable support during my PhD. Under his guidance I grew up to be a better researcher. Despite being the department chair, he has always made himself available for help and advice. Especially I would like to thank him for letting me patiently to explore a wide range of research. I have been extremely fortunate to have him as an advisor, mentor and a friend.

I want to thank my committee members Avrim Blum, John P. Dickerson, David Mount, and Louiqa Raschid for serving as my thesis committee. Avrim has been a great mentor, and his valuable insights, and his ability to explain difficult concepts intuitively has helped me a lot. I would also like to thank him for hosting me as an intern. John always makes himself available for help and advice to the students, and it was a pleasure to work with him. I am thankful to Barna Saha for hosting me for a visit to the University of Massachusetts Amherst. During that visit, I got interested into the problem of correlation clustering which eventually resulted into the Chapters 3 and 4. I would like to thank Alejandro A. Schaffer and Eytan Ruppin for introducing

me to the area of cancer research. All of this work would not have been possible without my other amazing collaborators, Faez Ahmed, Natalie Artzi, Hedyeh Beyhaghi, Mark Fuge, Sainyam Galhotra, Manish Purohit, Keziah Naggita, Roy Schwartz, Pattara Sukprasert, Rahulsimham Vegesna, and Sheng Yang. Special thanks to Niklas Karlossn for hosting me as an intern.

I am thankful to the other members of the theory group. Especially, I am thankful to my labmates, Ioana Bercea, Brian Brubach, Aounoun Kumar, Saurabh Kumar, Manish Purohit, and Pattara Sukprasert, and Sheng Yang for their friendship, and interesting discussions. I would like to convey my warm regards to the entire staff of the Computer Science department, in particular - Tom Hurst, Jodie Gray, Sharron McElroy, and Adelaide Findlay, for their support.

During the past two years, I was visiting the Computer Science Department at Northwestern University. I would like to thank the theory group at Northwestern University for their friendly and welcoming environment. I am thankful to the department staff at Northwestern University for their help during my visit.

A huge thanks to my beloved friends Ali, Alireza, Amin, Hadi, Hamed, Hamid, Hedyeh, Jerry, Kiana, Mahsa, Parsa, Saeed, Soheil for making some of the dearest of my memories during the past few years.

My deepest gratitude to my family Parvin, Asghar and Bahar for their everlasting love and support.

Table of Contents

List of Tables

List of Figures

1 Introduction

Algorithms are used ubiquitously in everyday life. They are used to make high-stakes decisions, for example, assessing credit score of a loan application, or evaluating job applicants. As a result, it is important to ensure these algorithms are compatible with social values. In this thesis, we take a step towards designing a few algorithmic paradigms while trying to address some key social concerns including fairness, diversity, and resilience to the strategic behavior of individuals.

Algorithmic fairness has gotten increasingly more important as more decisions are made by automated systems. Algorithms have been used traditionally for the purpose of credit decisions [1], and more recently for hiring decisions [2], and criminal justice sentencing [3]. Many automated algorithms were shown to have implicit biases against certain demographics [2]. Due to their growing impact on different aspects of everyday life, it is crucial to focus on designing algorithms that are inclusive, unbiased, and helpful to the entire

population.

Lack of diversity in team formation has a potential impact on discrimination against marginalized groups. When a homogenous team produces a technology, their outcome best serves a specific population. For example, Google searches have displayed less well-paid jobs to women compared to men [4]. Sometimes automated decision systems are made by learning from data that comes from society and has its own built-in biases. One way of mitigating inherent biases is to form a diverse team working on designing such algorithms to ensure our own privileged position does not make us completely blind to the experiences of others.

Another growing concern with respect to automated algorithms is their vulnerability to manipulation and gaming. When individuals have information about a decision rule, they may try to alter their attributes even artificially to achieve a better outcome. For example, consider the credit score system in the United States, where individuals participate in artificial actions like opening multiple credit lines in order to improve their credit score. As a result, it is important to design algorithms that are resilient to the strategic behavior of individuals.

In this dissertation we take an algorithmic approach towards increasing fairness, diversity, and resilience to the strategic behavior of individuals. The

following sections describe the problems that we consider and give a brief overview of our contributions and techniques.

1.1 Diverse Matching

Many applications of automated algorithms can be captured as a bipartite matching problem, e.g. healthcare, advertising, and general resource allocation. Ahmed et al. [5] introduced the notion of diverse bipartite b-matchings where the goal is to simultaneously maximize the efficiency of an assignment along with its diversity. For example, a firm might want to hire several highly skilled workers, but if that firm also cares about diversity it may want to ensure that some of those hires occur across marginalized categories of employees. They captured this problem as optimizing a submodular function over the matching.

In Chapter 2, we generalize the diverse matching problem and introduced matchings where each worker has multiple features (e.g., country of origin, gender) and our goal is to form diverse teams with respect to all these features. We show that unlike maximizing diversity along a single feature, the problem of simultaneously maximizing diversity along several features (e.g., country of citizenship, gender, skills) when the number of features is part of the input is

NP-hard. Next, we provide the first pseudo-polynomial time algorithm for the diverse bipartite b-matching with respect to multiple features problem with class-specific weights. That is, under conditions when the cost of assigning all items from one feature set to an item on the other side of the graph is the same. The key insight lies in detecting *negative cycles* in an auxiliary graph representation, which we use to either provide incremental improvements to the incumbent diverse matching or prove that our negative-cycle-detection algorithms have found a globally-optimal matching. We then extend the algorithm to the diverse bipartite b-matching problems with *general* edge weights, where edge weights of nodes within a category can be different. Lastly, we demonstrate our algorithm's applicability to a multi-aspect review assignment evaluation dataset [6], a benchmark dataset from UIUC, and show our algorithm takes less time to converge to an optimal solution than a proposed Mixed Integer Quadratic Programming approach (using a state-of-the-art commercial solver).

1.2 Fair Correlation Clustering

Many of the typical application scenarios where fairness has been identified to be crucial (e.g. lending, marketing, job selection etc.) requires clus-

tering large datasets with sensitive features. These datasets often come as a network. In order to incorporate fairness into clustering, the seminal work by Chierichetti et al. [7] proposed that each cluster has proportional representation from different demographic groups. They designed new approximation algorithms for the classic k-center and k-median clustering objectives with this notion of fairness. The fairness notion for k-center and k-median was further refined by Bercea et al. [8] and Bera et al. [9] by making sure that the demographic distribution of each cluster is the same as the global distribution of the demographics.

We considered a fair variant of the classic optimization problem of correlation clustering [10]. In the classic correlation clustering problem introduced by Bansal et al. [11], we are given a complete graph where each edge is labeled positive or negative. The goal is to obtain a clustering of the vertices that minimizes disagreements – the number of negative edges trapped inside a cluster plus positive edges between different clusters. Correlation clustering has many applications in social network analysis, data mining, computational biology, business and marketing [12–14]. In our work, our aim is to generate clusters with minimum disagreements, where the distribution of a feature (e.g. gender) in each cluster is same as the global distribution. For the case of two colors when the desired ratio of the number of colors in each cluster is 1 to p, we get

$\mathcal{O}(p^2)$-approximation algorithm. When there exists more than one color and the ratio of the number of nodes of color c_1 to color c_i is $1 : p_i$ ($\forall 1 < i \leq |\mathcal{C}|$), where $p_i \in \mathcal{Z}^{\geq 1}$, we propose an algorithm where the distribution of colors in each cluster is the same as the global distribution, and the total number of disagreements is at most $\mathcal{O}((\max_{i=1}^{|\mathcal{C}|}\{p_i\})^2 \cdot |\mathcal{C}|^2) \cdot OPT$. Next, we show how to avoid over-representation of any color in each cluster. For the case of two colors, given two ratios $1 : p$ and $1 : q$, we propose an algorithm that gives a clustering where the ratio of number of red nodes to blue nodes in each cluster is between $1 : q$ and $1 : p$, and the total number of disagreements is at most $\mathcal{O}(q^2) \cdot OPT$. When there exists more than one color and the ratio of the number of nodes of color c_1 to color c_i needs to be between $1 : q_i$ and $1 : p_i$ where $p_i, q_i \in \mathcal{Z}^{\geq 1}, p_i \leq q_i$, we propose an algorithm that gives a clustering where $\forall 1 < i \leq |\mathcal{C}|$, the ratio of number of nodes of color c_1 to color c_i in each cluster is between $1 : q_i$ and $1 : p_i$, and the total number of disagreements is at most $\mathcal{O}((\max_{i=1}^{|\mathcal{C}|}\{q_i\})^2) \cdot OPT$.

1.3 Min-max Correlation Clustering via MultiCut

In Chapter 4, we consider a different notion of group fairness for correlation clustering. We define a cluster-wise objective function that asks to minimize the maximum number of disagreements of each cluster. This captures the case when we wish to create communities that are harmonious, as minimizing the total number of disagreements could create an imbalanced community structure. This new local objective guarantees fairness to communities instead of individuals.

In recent work, Puleo and Milenkovic [15] introduced a local vertex-wise min-max objective for correlation clustering which bounds the maximum number of disagreements on each node. This problem arises in many community detection applications in machine learning, social sciences, recommender systems and bioinformatics [16–18]. This objective function makes sure each individual has a minimum quality within the clusters. They showed this problem is NP-hard even on un-weighted complete graphs, and developed an $O(1)$ approximation algorithm for unweighted complete graphs. Charikar et al. [19] improved upon the work by Puleo et al. [15] for complete graphs by giving a

7-approximation. For general weighted graphs, their approximation bound is $O(\sqrt{n})$ where n is the number of vertices. Kalhan et al. [20] later improved the bound for complete graphs to 5-approximation.

For the cluster-wise objective, we provide the first nontrivial approximation algorithm for this problem achieving an $\mathcal{O}(\sqrt{\log n \cdot \max\{\log(|E^-|), \log(k)\}})$ approximation for general weighted graphs, where $|E^-|$ denotes the number of negative edges and k is the number of clusters in the optimum solution. To do so, we also obtain a corresponding result for multicut where we wish to find a multicut solution while trying to minimize the total weight of cut edges on every component. The results are then further improved to obtain (i) $\mathcal{O}(r^2)$-approximation for min-max correlation clustering and min-max multicut for graphs that exclude $K_{r,r}$ minors (ii) a 14-approximation for the min-max correlation clustering on complete graphs.

1.4 The Strategic Perceptron

Automated classification systems make decisions based on perceived attributes of individuals. However, when individuals have information about the classifier, they may try to alter their attributes even artificially to achieve a better outcome. In Chapter 5, we consider the problem of learning a linear

classifier in presence of strategic agents. Consider the problem of learning a linear classifier when the data is unmanipulated and linearly separable, i.e. the feature space is divided into two half spaces with positive data points in one and negative data points in the other, and a nonzero margin between them. In this scenario, the Perceptron algorithm learns a linear classifier in a bounded number of mistakes. When individuals can manipulate, in each step, an individual with feature vector $\mathbf{z}$ that is not classified as positive, may prefer to pay a cost and pretend to have feature vector $\mathbf{x}$, if $\mathbf{x}$ gets classified as positive. The cost function can be either ℓ_2 or ℓ_1. In the former case, the cost is proportional to the Euclidean distance between $\mathbf{z}$ and $\mathbf{x}$. In the latter case, we assume agents can move along specific directions, and the total cost that needs to be paid for reaching from $\mathbf{z}$ to $\mathbf{x}$ is a weighted function of separate costs in each direction.

In Chapter 5, first we show that the original Perceptron algorithm fails to learn a linear classifier when individuals can manipulate their attributes even when a perfect classifier under manipulation exists. Next, we propose an algorithm robust to manipulation that finds a linear classifier in a bounded number of mistakes, for both scenarios when the cost of manipulation is ℓ_2 or weighted ℓ_1. Finally, we generalize our algorithm to the setting with unknown manipulation costs in the ℓ_2 case.

2 Diverse Matchings

2.1 Introduction

The bipartite matching problem occurs in many applications such as healthcare, advertising, and general resource allocation. Weighted bipartite b-matching is a generalization of this problem where each node on one side of the market can be matched to many items from the other side, and where edges may also have associated real-valued weights. Examples of weighted bipartite b-matching include medical interns or residents to hospitals [21], assigning children to schools [22, 23], reviewers to manuscripts [24, 25], and donor organs to patients [26, 27].

Ahmed et al. [28] introduced the notion of *diverse* bipartite b-matching, where the goal was to simultaneously maximize the "efficiency" of an assignment along with its "diversity." For example, a firm might want to hire several highly skilled workers, but if that firm also cares about diversity, it may want

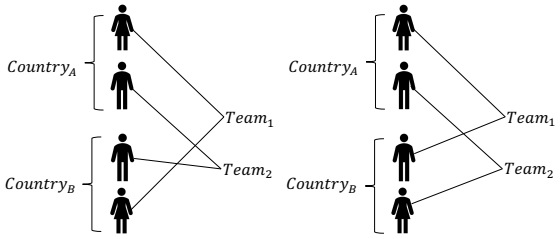


Figure 2.1: An illustrative example of single feature diverse matching (left) versus multi-feature diverse matching (right); here, the matching creates teams with workers from each country and gender.

to ensure that some of those hires occur across marginalized categories of employees. They proposed an objective which combined economic efficiency and diversity demonstrating that, in practice, reducing the efficiency of a matching by small amounts can often lead to significant gains in diversity across a matching. However, their formulation was limited to diversity for a single feature. It also relied on solving a general Mixed-Integer Quadratic Program (MIQP), which is flexible but computationally intractable.

In this work, we generalize the diverse matching problem and introduce matchings where each worker has *multiple* features (e.g., country of origin, gender) and our goal is to form diverse teams with respect to all these features. We found that the problem with a single feature, studied by Ahmed et al. [28], can be reduced to a minimum quadratic cost maximum flow formulation and solved in polynomial time by an existing algorithm designed by Minoux [29]. In contrast, we provide NP-hardness results for the general case of multiple

features.

Our contributions. Our main contributions are as following:

- We provide the first pseudo-polynomial time algorithm for the diverse bipartite b-matching w.r.t. multiple features problem with class-specific weights.[1] The key insight lies in detecting *negative cycles* in an auxiliary graph representation, which we use to either provide incremental improvements to the incumbent diverse matching or prove that our negative-cycle-detection algorithms have found a globally-optimal matching. We also provide a general MIQP formulation for this problem.
- We then extend the algorithm to the diverse bipartite b-matching problems with *general* edge weights, where edge weights of nodes within a category can be different.
- Lastly, we demonstrate our algorithm's applicability to paper-reviewer matching. Our algorithm takes less time to converge to an optimal solution than the proposed MIQP approach (using a state-of-the-art commercial solver).

[1] That is, under conditions when the cost of assigning all items from one feature set to an item on the other side of the graph is the same. This holds when, e.g., one is matching academic papers to reviewers where each reviewer can specify exactly one field of expertise and the cost of assigning a paper to any of the reviewers *within* the same field is the same but differs *across* fields.

2.2 Related Work

Matching people to form diverse teams leverages the intersection of two past areas of research: the role of team diversity in collaborative work and how diversity among groups of resources is measured and used to form/match teams. Compared to related work, our work provides a practical, high-performing method to perform diverse b-matching that can enable applications like diverse team formation or diverse resource allocation. Below we will use the example of diverse team formation (for example, in project teams within a company) to provide a concrete example to place prior work in context; however, our proposed approach is generally applicable to any diverse matching problem.

In the example of forming teams, the traditional approach is to use weighted bipartite b-matching (WBM) methods [30]. These methods maximize the total weight of the matching while satisfying some constraints. However, there are two major issues with these approaches. First, it assumes that the value provided by a person in a team is always fixed and independent of who else is in the team. This assumption may not hold in many cases. A new team member may provide more added value to the team if she is added to a smaller team compared to the case if she is added to a larger team. This property of diminishing marginal utility can be mathematically captured by a family of

functions called submodular functions, which we define later. Second, existing approaches do not account for diversity *within* a team, where teams with workers from different backgrounds may be desirable. For example, different types of worker diversity have a direct impact on the success rate of tasks [31]. Likewise, firms with a higher number of employees with higher education and diversity in the types of educations have a higher likelihood of innovating [32] and increasing revenue for firms [33]. In this chapter, we address both these issues.

Past researchers have generally measured diversity by defining some notion of *coverage*—that is, a diverse set is one that covers the space of available variation. Mathematically, researchers have done so via the use of *submodular functions*, which encode the notion of diminishing returns [34]; that is, as one adds items to a set that are similar to previous items, one gains less utility if the existing items in the set already "cover" the characteristics added by that new item. For example, many previous diversity metrics used in the information retrieval or search communities—including Maximum Marginal Relevance (MMR) [35] and Determinantal Point Processes [36]—are instances of submodular functions. These functions can model notions of coverage, representation, and diversity [37] and they have been shown to achieve top results on common automatic document summarization benchmarks—*e.g.*, at the Document

Understanding Conference [34].

Within matching, our work is closest to that of Ahmed et al. [28], which used a supermodular function to propose a diverse matching optimization method. Other researchers have also approached similar problems, with diversity either as an objective or as a constraint. For instance, Gölz and Procaccia [38] match migrants to localities in a way that maximizes the expected number of migrants who find employment. Benabbou et al. [39] study the trade-off between diversity and social welfare for the Singapore housing allocation. They model the problem as an extension of the classic assignment problem, with additional diversity constraints. Lian et al. [40] solve the assignment problem when preferences from one side over the other side are given and both sides have capacity constraints. They use order weighted averages to propose a polynomial-time algorithm which leads to high quality and more fair assignments. Agrawal et al. [41] show that a simple iterative proportional allocation algorithm can be tuned to produce maximum matching with high entropy. Finally, Kobren et al. [42] proposed two fairness-promoting algorithms for the paper-reviewer matching problem. They demonstrate that their algorithm achieves higher utility compared to state of the art matching algorithms that optimize for fairness only. In contrast, our goal is to develop an algorithm for finding the optimal assignment which maximizes utility as

well as diversity along multiple features as an objective—along with having constraints on workload.

We define a utility function that can be tuned to balance the diversity and total weight of matching. The diversity function is inspired by the Herfindahl index [43], which is a statistical measure of concentration and commonly used in economics. We provide a new algorithm that models the problem using an auxiliary graph and uses a heuristic improvement of the negative cycle detection of Bellman-Ford by Goldberg and Radzik [44][2] to find negative cycles and cancel them on a new graph to obtain an optimal solution for the original problem.

2.3 Preliminaries

In this section, we first define the preliminaries for a diverse matching problem, where workers are to be matched to teams and each team wants workers belonging to a diverse set of features. In our problem, we are given a set of features for the workers. Let $\mathcal{F} = \{f_1, \cdots, f_{|\mathcal{F}|}\}$ denote the feature set for the workers. An example of a feature set could be {country of citizenship, gender}. Each feature $f_k \in \mathcal{F}$ has one of the values $\mathcal{F}_k = \{f_{k,1}, \cdots, f_{k,|\mathcal{F}_k|}\}$.

[2]We used the negative cycle detection algorithm by [44]. Cherkassky et al. [45] compared the performance of multiple negative cycle detection algorithms, and the algorithm by Goldberg and Radzik [44] was one of the fastest.

Let $|\mathcal{F}_{k,k'}|$ denote the number of workers having value $f_{k,k'}$ for feature f_k. The set of workers is denoted by $X = \{x_1, \ldots, x_n\}$. X is partitioned into $|\mathcal{V}|$ subsets $V_1, \cdots, V_{|\mathcal{V}|}$, where each subset V_j corresponds to a feature set $v_j = \{v_{j,1}, \cdots, v_{j,|\mathcal{F}|}\}$, where $\forall 1 \leq k \leq |\mathcal{F}|, v_{j,k} \in \mathcal{F}_k$.

We wish to form a set of teams $\{T_1, \ldots, T_t\}$ of the workers where each team T_i has a demand of d_i, specifying the number of workers that need to be assigned to it. Each worker can be assigned to at most one team.

The diversity loss of an assignment is denoted by D and is equal to $\sum_{k=1}^{|\mathcal{F}|} \lambda_k D_k$, where D_k shows the diversity loss w.r.t. feature f_k, and $\lambda_k \in \mathcal{Z}^{\geq 0}$ is a constant. Let $c_{i,k,k'}$ denote the number of workers in T_i having value $f_{k,k'} \in \mathcal{F}_k$ for feature f_k. Then, $D_k = \sum_{i=1}^{t} \sum_{k'=1}^{|\mathcal{F}_k|} c_{i,k,k'}^2$. The cost of assigning each worker having value $f_{k,k'} \in \mathcal{F}_k$ for feature f_k to team T_i is denoted by $u_{i,k,k'} \in \mathcal{Z}^{\geq 0}$. We assume all costs are integers. The total cost of an assignment is $TU = \sum_{k=1}^{|\mathcal{F}|} TU_k$ where $TU_k = \sum_{i=1}^{t} \sum_{k'=1}^{|\mathcal{F}_k|} c_{i,k,k'} \cdot u_{i,k,k'}$.

Our goal is to minimize the objective function which is equal to $D + \lambda_0 TU$, where $\lambda_0 \in \mathcal{Z}^{\geq 0}$. To understand why minimizing D_k makes an assignment more diverse w.r.t f_k, consider Figure 2.1. In the left assignment, $D_2 = 8$, and in the right assignment $D_2 = 4$, and the right assignment is more diverse w.r.t f_2, i.e. gender. By setting λ parameters, we assume that the relative importance between factors is not qualitative and can be quantified. Next, we

provide Theorem 1, which shows that this problem is NP-hard.

Theorem 1. *Minimizing the supermodular diversity loss function w.r.t multiple features is NP-hard.*

Proof. First, we define a slight variation of 3-Color in the following:

OUR-3-Color: *Given a graph $G = (V, E)$ with n vertices, does there exist a coloring of all the vertices of G with 3 colors c_1, c_2, c_3 such that no two adjacent vertices receive the same color, and n_1 vertices are colored c_1, n_2 vertices are colored c_2, and n_3 vertices receive color c_3.*

In Lemma 1, we show *OUR*-3-Color is NP-hard by showing a reduction from 3-Color. For the sake of completeness, we define the problem 3-Color in the following:

3-Color: *Given a graph $G = (V, E)$ with n vertices, does there exist a coloring of vertices with 3 colors such that no two adjacent vertices receive the same color, and all the vertices are colored?*

Lemma 1. *Problem* OUR-3-Color *is NP-hard.*

Proof. We show a reduction from 3-Color to *OUR*-3-Color as following: Given an instance of 3-Color, consider all triples (n_1, n_2, n_3) such that $n_1 + n_2 + n_3 = n$, if for at least one triple, the answer to the *OUR*-3-Color problem is positive, so is the answer to the 3-Color. Since the number of triples is

$\Omega(n^3)$, the reduction is poly-time and therefore *OUR*-3-Color problem is NP-hard. □

Next, we show multiple-attribute diverse matching problem is NP-hard by showing a reduction from *OUR*-3-Color. Given an instance $(G(V, E), n_1, n_2, n_3)$ of *OUR*-3-Color, we define a feature f_e corresponding to each edge of $e \in E$, and a worker per each node of G, and look for a solution consisting of 3 teams with demands n_1, n_2, n_3 (each team corresponding to one color) that minimizes the objective function of multiple-attribute diverse matching problem. For each feature f_e corresponding to the edge $e \in E$, each worker is assigned a unique value for it, except the workers corresponding to the endpoints of e who get assigned the same value for this feature. Formally, assign a feature f_k to each edge $e_k = (v_{k_1}, v_{k_2}) \in E$. Let $f_{k,i}$ denote the value of f_k for the worker corresponding to $v_i \in V$. Let $f_{k,i} = i$ if $i \neq k_1, k_2$. Otherwise, let $f_{k,i} = 0$.

In the following, we show if the optimum solution of multiple-attribute diverse matching problem has value greater than some threshold (which is explained formally below), then it is clear that two workers having the same value for a feature are in the same team—which is equivalent to assigning the same color to the endpoints of an edge. Therefore, by looking at the optimum solution of multiple-attribute diverse matching problem we can realize whether *OUR*-3-Color instance is feasible or not.

To put it formally, consider an arbitrary edge $e_k(v_{k_1}, v_{k_2}) \in E$. If the workers corresponding to the endpoints of e_k belong to different teams, f_k contributes $n_1 + n_2 + n_3$ to the objective function since all the workers inside a team have different values for f_k. Otherwise, it contributes $n_1 + n_2 + n_3 - 2 + 2^2$ since workers corresponding to v_{k_1}, v_{k_2} are the only workers having the same value for f_k and they are assigned to the same team. If the cost of the optimal solution for the diverse matching problem is $(n_1 + n_2 + n_3) \cdot |E|$, then there does not exist a pair of workers in a team where the vertices corresponding to them are neighboring in G; which means in OUR-3-Color instance, no two adjacent vertices receive the same color, and the OUR-3-Color instance is feasible. Otherwise, if the cost of the optimal solution to the multiple-attribute diverse matching problem is more than $(n_1 + n_2 + n_3) \cdot |E|$, the OUR-3-Color instance is infeasible. □

We are interested in solving this NP-hard problem. We begin by presenting two different representations of instances of the problem: one in matrix form (used for expositional ease), and the other in graph form (used to build our optimal diverse matching algorithm in Section 2.4).

Matrix Representation. An example of matrix representation with three teams and two countries and two genders is shown in Table 2.1. Each column cor-

responds to a feature set and each row corresponds to a team. Entry $w_{i,j}$ shows the number of workers with feature set v_j assigned to T_i. We introduce a *dummy team* T_0, and $w_{0,j}$ shows the number of workers with feature set v_j who are not assigned to any team.

Matching Representation. In this representation, a bipartite graph $G = (\mathcal{X} \cup \mathcal{T}, E)$ is given. The nodes in $\mathcal{X}$ correspond to the workers, and are partitioned into $|\mathcal{V}|$ subsets where each subset corresponds to a feature set. Each vertex in $\mathcal{T}$ corresponds to a team in $\{T_0, T_1, T_2, \cdots, T_t\}$. The assignment of workers to teams forms a b-matching, where the degree of each node T_i for $1 \leq i \leq t$ is d_i. All workers who are not assigned to any team $T_1, \cdots, T_t$ get assigned to the dummy team T_0. Degree of each node $x \in \mathcal{X}$ is exactly one.

Local Exchange. A local exchange happens when a group of teams decides to transfer one or more workers between each other while maintaining the total number of workers in each of them. The exchange is done in a way that the initial demands of all the teams are fulfilled. Arrows in Table 2.1 show a local exchange in a matrix representation.

In this exchange, one worker from V_2 is moved from T_3 to T_1. Two workers from V_1 are moved. One is moved from T_1 to T_2, and the other one is moved from T_2 to T_3. The set of edges of local exchange in a matrix

	$country_1, g_1$	$country_1, g_2$	$country_2, g_1$	$country_2, g_2$
T_0	$w_{0,1}$	$w_{0,2}$	$w_{0,3}$	$w_{0,4}$
T_1	$w_{1,1}$	$w_{1,2}$	$w_{1,3}$	$w_{1,4}$
T_2	$w_{2,1}$	$w_{2,2}$	$w_{2,3}$	$w_{2,4}$
T_3	$w_{3,1}$	$w_{3,2}$	$w_{3,3}$	$w_{3,4}$

Table 2.1: Matrix representation of three teams and workers from two countries and two genders. Dummy team T_0 accommodates unassigned workers. Arrows represent a local exchange.

	$country_1$	$country_2$
T_0	$c_{0,1,1} = w_{0,1} + w_{0,2}$	$c_{0,1,2} = w_{0,3} + w_{0,4}$
T_1	$c_{1,1,1} = w_{1,1} + w_{1,2}$	$c_{1,1,2} = w_{1,3} + w_{1,4}$
T_2	$c_{2,1,1} = w_{2,1} + w_{2,2}$	$c_{2,1,2} = w_{2,3} + w_{2,4}$
T_3	$c_{3,1,1} = w_{3,1} + w_{3,2}$	$c_{3,1,2} = w_{3,3} + w_{3,4}$

Table 2.2: Matrix representation embedding w.r.t country.

representation is called a cycle. The source-transitions of a cycle are the cells without any input edges, and the sink-transitions are the cells without any output edges. In Table 2.1, the cells corresponding to $w_{3,2}$ and $w_{1,1}$ are source-transitions, and the cells corresponding to $w_{1,2}$ and $w_{3,1}$ are sink-transitions.

Figure 2.2 shows the same local exchange operation using a matching representation. In this figure, the black matching shows the initial assignment, and the dotted matching shows the assignment after the exchange operation is done.

	g_1	g_2
T_0	$c_{0,2,1} = w_{0,1} + w_{0,3}$	$c_{0,2,2} = w_{0,2} + w_{0,4}$
T_1	$c_{1,2,1} = w_{1,1} + w_{1,3}$	$c_{1,2,2} = w_{1,2} + w_{1,4}$
T_2	$c_{2,2,1} = w_{2,1} + w_{2,3}$	$c_{2,2,2} = w_{2,2} + w_{2,4}$
T_3	$c_{3,2,1} = w_{3,1} + w_{3,3}$	$c_{3,2,2} = w_{3,2} + w_{3,4}$

Table 2.3: Matrix Representation Embedding w.r.t to gender.

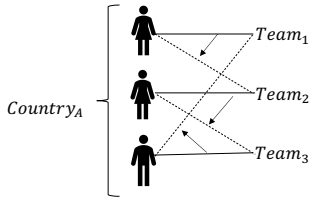


Figure 2.2: Local exchange operation (in matching representation).

Gain of a local exchange. Our goal is to minimize the objective function f, by doing some local exchanges. To find out, we first calculate the marginal gain from a given exchange operation which is the difference between the objective values before and after a local exchange. In order to simplify this concept, we use the following definition:

Embedding of Matrix Representation. Consider a given matrix representation M, it can be embedded into a matrix M_k for a fixed feature f_k in the following way: all the columns in M corresponding to the same value $f_{k,k'}$ of f_k, are combined into a single column in M_k. For example, embedding of the matrix representation in Table 2.1 into M_1, M_2 w.r.t. the features country and gender are shown in Tables 2.2 and 2.3. Since in M_1, the

number of people assigned from each country to each team is not changed, $\Delta_1 = \Delta(\lambda_0 \cdot TU_1 + \lambda_1 D_1) = 0$. According to M_2, $\Delta_2 = \Delta(\lambda_0 \cdot TU_2 + \lambda_2 D_2) = \lambda_0\big(- u_{3,2,2} + u_{1,2,2} - u_{1,2,1} + u_{3,2,1}\big) + \lambda_2\big((c_{3,2,2} - 1)^2 - (c_{3,2,2})^2 + (c_{1,2,2} + 1)^2 - (c_{1,2,2})^2 + (c_{1,2,1} - 1)^2 - c_{1,2,1}^2 + (c_{3,2,1} + 1)^2 - c_{3,2,1}^2\big)$.

It can be seen that the contribution of the cells which are not source-transition or sink-transition to the gain of a local exchange is zero (all the cells in the local exchange in Table 2.2, and the node corresponding to $c_{2,2,1}$ in M_2). If the net gain, i.e. $\Delta_1 + \Delta_2$, is negative, then the local exchange can be considered beneficial, and we can transfer the workers.

2.4 Negative-Cycle-Detection-based Algorithms

In this section, we explain our algorithm for finding the optimum assignment. First, we build an auxiliary graph G'. For each team $T_i \in \{T_0, \cdots, T_t\}$, there is a switch in G' with $|\mathcal{V}|$ input ports, and $|\mathcal{V}|$ output ports. Each port is a node in G', and each switch is a directed bipartite graph, with edges going from its input ports (nodes) to its output ports. In Figure 2.3, each box is a switch. Inside a switch T_i, there is a directed edge from each input port to each output port. If the directed edge is connecting two ports such that their

corresponding combinations of features do not have the same value for any features, the weight of this edge is equal to zero. Otherwise, per each feature f_k that has the same value, $-2\lambda_k$ is added to the weight of this edge.

The reason behind assigning these weights to the edges is to make sure in a local exchange, considering a fixed feature f_k, the cells which are not a source-transition or a sink-transition w.r.t. M_k, have zero contribution to $\Delta(D_k)$.

For each pair of teams T_{i_1} and T_{i_2} where $i_1 \neq i_2$, and for each feature combination v_j, there is a directed edge from output port $O_j^{i_1}$ of switch T_{i_1} to the input port $I_j^{i_2}$ of switch T_{i_2}, and weight of this edge captures the difference in the objective function when in the matrix representation a person in column V_j (with feature set v_j) is moved from T_{i_1} to $T_{i,2}$.

Each cycle in this graph corresponds to a cycle in a matrix representation and local exchanges along them have the same gain. Figure 2.3 shows a cycle which is corresponding to the cycles in Table 2.1 and Figure 2.2.

After constructing the auxiliary graph, we run Algorithm 1. Algorithm 1 moves workers from one team to another if it detects a negative cycle.

Algorithm 1 takes as input an initial feasible solution Q as input. To find Q, we first find a feasible solution, which satisfies all the demand constraints. In order to find an initial feasible solution, in each iteration, consider the first

Algorithm 1: Find optimal diverse b-matching

Input : Directed weighted graph G', initial feasible b-matching Q which satisfies team demands.

Output: Optimal diverse b-matching

while $\exists$ *a negative cycle* $C \in G'$ **do**

  // Perform a local exchange operation along C;

  for $e \in C$ **do**

    // Assume edge e is from output port $O_j^{i_1}$ of team T_{i_1} to input port $I_j^{i_2}$ of another team T_{i_2};

    // Move one worker with feature set $v_j = \{f_{1,k'_1}, \cdots, f_{|\mathcal{F}|,k'_{|\mathcal{F}|}}\}$ from team T_{i_1} to team T_{i_2}:

    $\forall k \in \{1, \cdots, |\mathcal{F}|\}$: $c_{i_1,k,k'_k} -= 1, c_{i_2,k,k'_k} += 1$;

    Update weight of edges of G' w.r.t to the new values of c_{i_1,k,k'_k}, and c_{i_2,k,k'_k};

subset of workers in the the bipartite graph G (V_j) with at least one un-assigned worker, and the first team (T_i) such that the number of workers assigned to it is less than its demand (In the first iteration, we start with V_1, T_1, and all the workers are un-assigned). Assign un-assigned workers from V_j to T_i, until either demand of T_i is fully satisfied, in this case, move to the next team ($i = i + 1$), or all the workers from V_j are assigned, then let $j = j + 1$. Repeat this procedure until all the demand constraints are satisfied. Time complexity of this procedure is $\mathcal{O}(|\mathcal{V}| + t)$.

In Algorithm 1, any negative cycle detection algorithm can be used to detect negative cycles in G'. We use a heuristic improvement of Bellman-Ford proposed by Goldberg and Radzik [44] in our experiments.

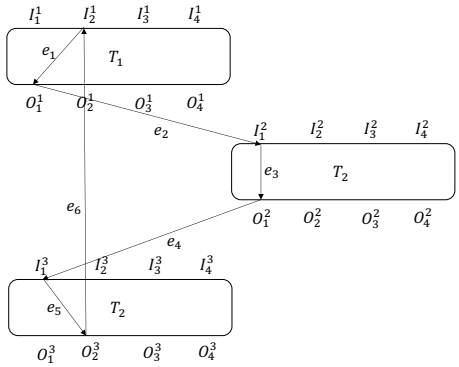


Figure 2.3: A local Exchange in graph representation.

2.5 Proof of Optimality

In this section, we prove that Algorithm 1 gives the optimum solution for diverse bipartite b-matching problem.

Assume after the algorithm ends, the final assignment is a local optimum P, and the optimum solution is P^*. Consider the matching representations of P and P^*. Since all the nodes in P and P^* are matched, the symmetric difference of P and P^* ($P \oplus P^*$) can be decomposed into a set of alternating even cycles. Each local exchange along an alternating cycle corresponds to a cycle in the matrix representation.

Before proving Thm. 2, we need the following definitions:

Maximal Cycle. A cycle y in a matrix representation M is maximal if its source-transitions and sink-transitions are source-transition and sink-transition

	$country_1, g_1$	$country_1, g_2$	$country_2, g_1$	$country_2, g_2$
T_0	$w_{0,1}$	$w_{0,2}$	$w_{0,3}$	$w_{0,4}$
T_1	$w_{1,1}$	$w_{1,2}$	$w_{1,3}$	$w_{1,4}$
T_2	$w_{2,1}$	$w_{2,2}$	$w_{2,3}$	$w_{2,4}$
T_3	$w_{3,1}$	$w_{3,2}$	$w_{3,3}$	$w_{3,4}$

Table 2.4: Maximal cycle decomposition

w.r.t all the edges in M as well. For example, consider Table 2.4. Let's call the dotted cycle y_g, the dashed cycle y_r, and the solid-line cycle y_b. y_g has two source-transitions $w_{1,1}, w_{0,3}$, and it has two sink-transitions $w_{0,1}, w_{1,3}$. Since there are no edges going out of $w_{1,3}, w_{0,1}$, and no edges going into $w_{0,3}, w_{1,1}$, y_g is a maximal cycle. Cycles y_r, y_b are maximal cycles as well. Therefore, $\{y_g, y_b, y_r\}$ gives a maximal cycle decomposition for M. However, if we consider embedding of M w.r.t gender (M_2), then y_r is not a maximal cycle anymore, and $\{y_g, y_r \cup y_b\}$ gives a maximal cycle decomposition w.r.t M_2 and M_1(embedding w.r.t countries). A cycle is called all-maximal cycle if it is maximal w.r.t all the matrix representations $M_1, \cdots, M_{|\mathcal{F}|}$. In this example, $\{y_g, y_r \cup y_b\}$ gives an all-maximal cycle decomposition.

Lemma 2. *The set of all the edges of $P \oplus P^*$ can be decomposed into a set of all-maximal cycles.*

Proof. Consider an arbitrary decomposition of the edges of $P \oplus P^*$ in the

matrix representation into a set of cycles $\{y_1, \cdots, y_\ell\}$. If there exists a cycle in $P \oplus P^*$ without any source-transitions and sink-transitions, it means the gain of this cycle is zero and it could be discarded. If there exists any cycle y_p which is not all-maximal, then there exists another cycle y_q which makes y_p not to be maximal w.r.t some features. For example in Figure 2.4, y_r is not maximal because of y_b. In this case, union y_p and y_q, and make $y_p \cup y_q$ a single cycle in the decomposition. At the end, all the edges in $P \oplus P^*$ will be decomposed into a set of all-maximal cycles. Let's call the set of all-maximal cycles $\{y'_1, \cdots, y'_{\ell'}\}$. □

Theorem 2. *Algorithm 1 finds the global optimum for the diverse b-matching problem.*

Proof. Let $f(P)$ show the value of the objective function for the assignment P. $f(P^*) - f(P) < 0$ therefore:

$$f(P^*) - f(P) = gain(y'_{1,1}) + gain(y'_{2,2}) + \cdots + gain(y'_{\ell',\ell'}) < 0$$

Where y'_k $(1 \leq k \leq \ell')$ is the k^{th} cycle in the all-maximal cycle decomposition, and $y'_{k,k}$ is applying the local exchange of the cycle y'_k at step k. The initial step is the assignment P. Since $f(P^*) - f(P) < 0$, there must be a maximal cycle y'_g such that $gain(y'_{g,g}) < 0$. We wish to show $gain(y'_{g,1}) < 0$, which implies starting from the initial assignment P, a local exchange can be done

with a negative gain, and P is not a local optimum which is a contradiction.

Let $D(y'_{g,g}), U(y'_{g,g})$ denote respectively the change in the diversity loss, and the change in the utility when applying a local exchange y'_g in step g. Let $D_{f_k}(y'_{g,g}), U_{f_k}(y'_{g,g})$ denote the change in the diversity loss w.r.t the feature f_k, when applying $y'_{g,g}$. Therefore:

$$gain(y'_{g,g}) = \sum_{k \in |\mathcal{F}|} \Big(D_{f_k}(y'_{g,g}) + U_{f_k}(y'_{g,g}) \Big)$$

Lemma 3 shows if $D_{f_k}(y'_{g,g}) < 0$, then $D_{f_k}(y'_{g,1}) < 0$. As a result, $D(y'_{g,g}) < 0$ implies $D(y'_{g,1}) < 0$. It is easy to see that $U(y'_{g,g}) = U(y'_{g,1})$. Therefore, $gain(y'_{g,g}) < 0$ implies $gain(y'_{g,1}) < 0$, and the proof is complete. □

Lemma 3. *If $D_{f_k}(y'_{g,g}) < 0$, then $D_{f_k}(y'_{g,1}) < 0$.*

Proof. Consider $y'_{g,g}$ embedded into M_k. There are four types of vertices in $y'_{g,g}$:

- Vertices in the form of $w_{0,j}$ where $1 \leq j \leq |\mathcal{V}|$. These vertices have contribution zero to both $D_{f_k}(y'_{g,g})$ and $D_{f_k}(y'_{g,1})$.
- Vertices that are not sink-transition or source-transition, i.e. $w_{2,2}$ in Figure 2.1, w.r.t M_k. It could be seen that contribution of these nodes to both $D_{f_k}(y'_{g,g})$ and $D_{f_k}(y'_{g,1})$ is zero.
- Sink-transitions: Consider an arbitrary sink-transition v in $y'_{g,g}$. Assume

the value of this node at the beginning of step g is v_g. The contribution of v to $D_{f_k}(y'_{g,g})$ is $\lambda_k\big((v_g+1)^2 - v_g^2\big) > 0$. Since v is a sink-transition, $v_g \geq v_1$. Therefore, $\lambda_k\big((v_g+1)^2 - v_g^2\big) \geq \lambda_k\big((v_1+1)^2 - v_1^2\big)$.

- Source-transitions: Consider an arbitrary source-transition v in $y'_{g,g}$. The contribution of v to $D_{f_k}(y'_{g,g})$ is $\lambda_k\big((v_g-1)^2 - v_g^2\big)$. Since v is a source-transition $v_1 \geq v_g$, and therefore $\lambda_k\big((v_g-1)^2 - v_g^2\big) \geq \lambda_k\big((v_1-1)^2 - v_1^2\big)$.

At the end, contribution of all the vertices to $D_{f_k}(y'_{g,1})$ is upper bounded by their contribution to $D_{f_k}(y'_{g,g})$. Therefore if $D_{f_k}(y'_{g,g}) < 0$, then $D_{f_k}(y'_{g,1}) < 0$. □

Theorem 3. *The running time of the algorithm is* $\mathcal{O}((\lambda_{\max} \cdot |\mathcal{F}| \cdot n^2 + \lambda_0 U) \cdot |\mathcal{V}|^2 \cdot t^2(|\mathcal{V}|+t))$, *where* U *is the maximum cost of an initial feasible b-matching and* $\lambda_{max} = \max\{\lambda_1, \cdots, \lambda_{|\mathcal{F}|}\}$.

In order to prove this theorem, first we show the following lemmas hold.

Lemma 4. *The number of iterations of our algorithm is at most* $\lambda_{max} \cdot |\mathcal{F}| \cdot n^2 + \lambda_0 U$.

Proof. The initial state of the algorithm is a feasible b-matching with cost at most U. Diversity loss of any matching is at most $\lambda_{max} \cdot |\mathcal{F}| \cdot n^2$. At each iteration, we find a negative weight cycle and since all the weights are integers,

its weight can be at most -1. Therefore, the objective function decreases by at least 1 at each step, and since the value of the objective function is always positive, the number of iterations is at most $\lambda_{max} \cdot |\mathcal{F}| \cdot n^2 + \lambda_0 U$. □

Lemma 5. *The complexity of each iteration of the algorithm is* $\mathcal{O}(|\mathcal{V}|^2 \cdot t^2(|\mathcal{V}| + t))$.

Proof. At each iteration, we use a negative cycle detection algorithm with running time $\mathcal{O}(|V| \cdot |E|)$ (where $|V|$ is the number of nodes in the auxiliary graph and $|E|$ is the number of edges). The number of nodes in the graph is $2|\mathcal{V}| \cdot (t+1)$, since there are $t+1$ switches in the graph and each switch has exactly $2|\mathcal{V}|$ ports and each port is a node in the graph. The number of edges incident on each port is $|\mathcal{V}| + t$. Therefore, the total number of edges is $\mathcal{O}(|\mathcal{V}| \cdot t(|\mathcal{V}| + t))$. Hence, the complexity of each iteration is $\mathcal{O}(|\mathcal{V}|^2 \cdot t^2(|\mathcal{V}| + t))$. □

Combining Lemma 4 with Lemma 5, and considering $\mathcal{O}(|\mathcal{V}| + t)$ time complexity for finding an initial feasible solution, yields Theorem 3.

2.6 Diverse Weighted Bipartite b-Matching

In this section, we extend our algorithm to solve the case where the cost of assigning workers from the *same* feature set to a team can be different. First, in each switch we put input and output ports for each worker. Inside each switch,

there is a complete bipartite graph from input ports to the output ports. Consider an edge between an input port to an output port corresponding to workers x_i and x_j. Per each feature f_k where x_i, x_j have the same values for f_k, $-2\lambda_k$ is added to the weight of the edge between x_i, x_j.

Consider an edge from output port $x_k^{i_1}$ of switch T_{i_1} to input port $x_k^{i_2}$ of switch T_{i_2}, where $x_k \in V_j$. The weight of this edge is equal to the change in the objective function by moving one worker from V_j out of T_{i_1}, and adding that worker to T_{i_2}. The proof of the following theorem is similar to Thm. 3.

Theorem 4. *The running time of the algorithm for general weights is* $\mathcal{O}((\lambda_{max} \cdot |\mathcal{F}| \cdot n^2 + \lambda_0 U) \cdot n^2 \cdot t^2(n+t))$*, where* U *is the maximum cost of any feasible b-matching.*

2.7 Experimental Validation & Discussion

To demonstrate the effectiveness of the proposed method, we apply it to a dataset of reviewer paper matching. First, we find the optimal solution for multi-feature reviewer paper matching and compare it to the single feature diverse matching method. We also provide the MIQP formulation of the same

problem based on literature and show how our algorithm is faster to the Gurobi based MIQP solver.

For the reviewer assignment problem, where each reviewer has multiple features, we want to match each paper with reviewers who are not only from different expertise areas (clusters), but also belong to different genders. We use the multi-aspect review assignment evaluation dataset [6], a benchmark dataset from UIUC. It contains 73 papers accepted by SIGIR 2007, and 189 prospective reviewers who had published in the main information retrieval conferences. The dataset provides 25 major topics and for each paper in the set, an expert provided 25-dimensional label on that paper based on a set of defined topics. Similarly for the 189 reviewers, a 25-dimensional expertise representation is provided.

To compare our method (Algorithm 1) with a baseline, we formulate a multi-feature MIQP variant of our problem, which is an extension of the single-feature formulation provided in [28] and is given by:

$$\min \lambda_0 \sum_{k=1}^{|\mathcal{F}|} \sum_{i=1}^{t} \sum_{k'=1}^{|\mathcal{F}_k|} u_{i,k,k'} \cdot c_{i,k,k'} + \sum_{k=1}^{|\mathcal{F}|} \lambda_k \sum_{i=1}^{t} \sum_{k'=1}^{|\mathcal{F}_k|} c_{i,k,k'}^2$$

$$\sum_{k=1}^{|\mathcal{F}|} \sum_{k'=1}^{|\mathcal{F}_k|} c_{i,k,k'} = d_i, \forall 1 \leq i \leq t$$

$$\sum_{i=0}^{t} c_{i,k,k'} = |\mathcal{F}_{k,k'}|, 1 \leq k \leq |\mathcal{F}|, 1 \leq k' \leq |\mathcal{F}_k|$$

To set up the graph for our method, we first cluster the reviewers into 5 clusters based on their topic vectors using spectral clustering. To calculate the relevance of each cluster for any paper, we take the average cosine similarity of label vectors of reviewers in that cluster and the paper. We set the constraints such that each paper matches with exactly 4 reviewers, and no reviewer is allocated more than 1 paper. To increase dataset size, we double the number of reviewers by creating a copy of each reviewer. As the original dataset lacks gender information, we added a new feature to each reviewer in this dataset by randomly adding one of two gender labels (Male or Female) to each reviewer. We set $\lambda_0 = \lambda_1 = \lambda_2 = 1$ for our experiments. Note that by varying these parameters, one can create the Pareto optimal frontier too.

We run the negative cycle detection algorithm, and the MIQP solver using Gurobi to find the optimum solution. On converging to the optimal solution, we find that all 73 papers receive two male reviewers and two female reviewers, which shows that the method was capable of balancing gender diversity. Each paper receives reviewers from four different clusters. If we only optimize for cluster diversity, it is possible that the gender ratio for individual paper gets skewed. When we run the same model with $\lambda_g = 0$ (no weight to gender diversity), we find that out of 73 papers, 12 papers receive all four reviewers of the same gender and 41 papers receive three reviewers of the same

gender. Hence, only 27.3% teams of reviewers are gender balanced. However, one should note that when we do not keep gender as an objective, the resultant allocation is random and different skewness can be observed in different runs based on the initial solution.

Finally, we compare the timing performance of our algorithm with MIQP by changing the number of papers that need to be reviewed on a Dell XPS 13 laptop with i7 processor. For MIQP, we set a maximum run time of four hours (14400 seconds) for Gurobi solver, at which we report the current best MIQP solution. Table 2.5 shows that for all cases with the number of papers greater than 13, MIQP does not converge within four hours, while our method finds the optimum solution in lesser time. Interestingly, MIQP current solutions are found to be the same as the optimum solution found by our method, which shows that for this application, MIQP was able to search the solution but it was not able to prove that the solution is optimum. In contrast, our method finds the solution faster as well as guarantees that it is optimum.

2.8 Authors

This Chapter was written by Saba Ahmadi, Faez Ahmed, John P. Dickerson, Mark Fuge, and Samir Khuller. It was published at the International

# Papers	# Reviewers	MIQP Time (s)	Our Method Time (s)
03	378	24.68	0.18
13	378	3979.90	14.84
23	378	14400.00	122.96
33	378	14400.00	400.56
43	378	14400.00	825.95
53	378	14400.00	2837.15
63	378	14400.00	5453.58
73	378	14400.00	11040.55

Table 2.5: Comparison of MIQP and our method for UIUC reviewer dataset with each paper needing 4 reviewers.

Joint Conferences on Artificial Intelligence (IJCAI) 2020 [46].

3 Fair Correlation Clustering

3.1 Introduction

The ubiquitous use of Machine learning tools for everyday decision making has brought the issue of fairness to the forefront. Many automated algorithms were shown to have implicit biases against certain demographics. In order to build machine learning algorithms that are inclusive, unbiased and helpful to the entire population, the recent years have seen a surge of research related to *fairness*. Many of the typical application scenarios where fairness has been identified to be crucial (e.g. lending, marketing, job selection etc.) requires clustering large datasets with sensitive features. These datasets often come as a network. In order to incorporate fairness into clustering, the seminal work by Chierichetti, Kumar, Lattanzi, and Vassilvitskii [7] incorporated a notion of group fairness into k-center and k-median clustering algorithms. Their goal was to form clusters such that given two groups, no group was

over-represented or under-represented in any cluster. The fairness notion was further refined by Bercea et al. [47] and Bera et al. [9] by making sure that the demographic distribution of each cluster is the same as the global distribution of the demographics. Subsequently, Schmidt, Schwiegelshohn, and Sohler [48] extended the framework to k-means clustering. While these works study clustering algorithms over a metric space, many clustering applications work over network data, and that calls for designing *graph clustering* algorithms that are fair to all demographics. In a very recent work, Kleindessner, Samadi, Awasthi and Morgenstern [49] consider the problem of fair spectral clustering . They prove rigorous theoretical bounds for their algorithms over the stochastic block model. However, the analysis over arbitrary networks is still not known. In particular, the use of triangle inequalities makes the analysis of metric based fair clustering easier compared to graph clustering where the metricity is lacking.

In this paper, we consider a fair variant of the classic optimization problem of correlation clustering. Correlation clustering is one of the most widely used clustering paradigms, and as claimed by Bonchi et al. [13] "arguably the most natural formulation of clustering". Given a set of objects and a pairwise similarity measure between them, the objective is to partition the objects so that, to the best possible extent, similar objects are put in the same cluster

and dissimilar objects are put in different clusters. This is represented by constructing a complete graph where edges are either labeled positive (similar objects) or negative (dissimilar objects). The edges can also be weighted. An algorithm for correlation clustering aims to minimize the disagreements among vertices, calculated as the weight of negative edges trapped inside a cluster plus positive edges between different clusters. As it just requires a definition of similarity, it can be applied broadly to a wide range of problems in different contexts such as social network analysis, data mining, computational biology, business and marketing [12–14].

Similar to other clustering algorithms, the known algorithms for correlation clustering may produce significantly biased output. In this work, we study a fair variant of the correlation clustering problem where each vertex has a given color, and the goal is to make sure that the distribution of the colors is the same as the global distribution in each cluster. This is the same notion of fairness studied by Bercea et al. [47] and Bera et al. [9] on k-center and k-median. In another variation, the goal is to make sure the number of nodes of a specific feature c_i in a cluster of size n is between $\frac{n}{q_i}, \frac{n}{p_i}$ where $p_i \leq q_i \in \mathcal{Z}^{\geq 1}$, and p_i, q_i are specified per each feature c_i. This later model was originally proposed by Ahmadian, Epasto, Kumar and Mahdian [50] with only the upper bound and later Bera et al. [9] generalized it to consider

lower bound (both p_i and q_i). Having a lower bound ensures that every color is represented in each cluster. They studied the k-center problem under this framework. In all our algorithms, we maintain the fairness constraints strictly, and optimize the objective function. That is, we give *exact* approximation algorithms, as opposed to *bi-criteria* approximation.

In a parallel independent work, Ahmadian et al. [51], study the problem of fairness in correlation clustering. Their work considers an upper bound constraint on the number of nodes of each color (e.g. $1 : t$) as opposed to considering both lower and upper bounds. Additionally, our algorithm guarantees identifying clusters that satisfy the fairness constraints exactly as opposed to a bi-criterion solution of Ahmadian et al. [51]. In their approach, they may violate the desired ratio of each color in each cluster by a factor of 2. Their approach does not violate the fairness constraints in the following two special cases, where the desired number of vertices of each color is equal in every cluster.

- For two colors Ahmadian et al. [51] achieve 256-approximation as opposed to 10.18-approximation in our work (Theorem 5).
- For more than two colors (say C), Ahmadian et al. [51] and our algorithm has $O(C^2)$ approximation (Corollary 1).

Contributions & Roadmap

Our contributions are as follows.

For the first fairness variant, we can assume that each node has a color, and the goal is to keep the distribution of the colors in each cluster same as the global distribution. First, we show our results for the case of 2 colors, and later we extend the results to an arbitrary number of colors. Our approach for 2 colors has some similarities to the approach proposed by Chierichetti et al. [7] for the k-center and k-median problems. The analysis of fair clusters with centroid based objectives leverages triangle inequality to bound the total objective value of the returned clusters. However, calculating the total disagreements for correlation clustering requires us to analyze the graph properties, leading to a completely different analysis as compared to [7].

Assume nodes in the input graph are either red or blue and the goal is to have a ratio of $1 : p$ of the number of red nodes to the number of blue nodes, where $p \in \mathcal{Z}^{\geq 1}$. In Section 3.5, we design a new algorithm with the following guarantees:

Theorem 6 (Two Colors)**.** *Given a complete unweighted graph $G = (V, E)$ where edges are labeled positive or negative, and the nodes are either red or blue where the ratio of number of red nodes to the number of blue nodes is*

$1 : p$ *for* $p \in \mathcal{Z}^{\geq 1}$*, there exists an algorithm which gives a clustering with ratio* $1 : p$ *of number of red to blue nodes in each cluster and at most* $\mathcal{O}(p^2) \cdot OPT$ *disagreements.*

Before explaining the general result for handling a ratio of $1 : p$, in Section 3.4, we explain a simpler scenario where the desired ratio of red to blue in each cluster is $1 : 1$. In this section, the following theorem is proved:

Theorem 5 (Warm-up)**.** *Given a complete unweighted graph* $G = (V, E)$ *where edges are labeled positive or negative, and the nodes are either red or blue, with an equal number of red and blue nodes, there exists an algorithm which gives a clustering with equal number of red and blue nodes in each cluster and at most* $(3\alpha + 4) \cdot OPT$ *disagreements, where* α *is the best approximation ratio for correlation clustering on a complete unweighted graph with minimizing disagreements objective.*[1]

In Section 3.5, we generalize our results and prove the following theorem:

Theorem 7 (Multiple Colors)**.** *Given a complete unweighted graph* $G = (V, E)$ *where edges are labeled positive or negative, and each node has exactly one of the colors* $\{c_1, \cdots, c_{|\mathcal{C}|}\}$*, and the ratio of the number of nodes of color* c_1 *to*

[1]The best known value of $\alpha = 2.06$, giving 10.18-approximation of our algorithm.

color c_i is $1 : p_i$ ($\forall 1 < i \leq |\mathcal{C}|$), where $p_i \in \mathcal{Z}^{\geq 1}$, there exists an algorithm where the distribution of colors in each cluster is the same as the global distribution, and the total number of disagreements is at most $\mathcal{O}((\max_{i=1}^{|\mathcal{C}|}\{p_i\})^2 \cdot |\mathcal{C}|^2) \cdot OPT$.

For a special case where $p_i = 1, \forall i$, we get the following Corollary.

Corollary 1 (Multiple Colors)**.** *Given a complete unweighted graph $G = (V, E)$ where edges are labeled positive or negative, and each node has exactly one of the colors $\{c_1, \cdots, c_{|\mathcal{C}|}\}$, and equal number of nodes of each color, there exists an algorithm such that the total number of disagreements is at most $\mathcal{O}(|\mathcal{C}|^2) \cdot OPT$.*

In Section 3.5, we prove NP-hardness of fair correlation clustering problem on complete unweighted graphs even for 2 colors. Note that the hardness result does not directly follow from the hardness result of the original correlation clustering.

In Section 3.5, we consider the fairness model studied by Ahmadian et al. [50] for k-center problem without over-representation. We are inspired by their definition of over-representation, and show the following theorem holds:

Theorem 8 (Avoiding Over-Representation)**.** *Given a complete unweighted graph $G = (V, E)$ where edges are labeled positive or negative, and nodes are colored red or blue, and two ratios $1 : p, 1 : q$ where $p, q \in \mathcal{Z}^{\geq 1}, p \leq q$, where*

ratio of the total number of red nodes to the total number of blue nodes is between $1:q$ *and* $1:p$*, there exists an algorithm which gives a clustering where the ratio of number of red nodes to blue nodes in each cluster is between* $1:q$ *and* $1:p$*, and the total number of disagreements is at most* $\mathcal{O}(q^2) \cdot OPT$*.*

We extend Theorem 8 to the case with multiple colors.

Theorem 9 (Avoiding Over-Representation with Multiple Colors)**.** *Given a complete unweighted graph* $G = (V, E)$ *where edges are labeled positive or negative, and each node has exactly one of the colors* $\{c_1, \cdots, c_{\mathcal{C}}\}$*, and two ratios* $1:p_i, 1:q_i$ *for each color* c_i *where* $p_i, q_i \in \mathcal{Z}^{\geq 1}, p_i \leq q_i$*, where ratio of the total number of nodes of color* c_1 *to the total number of nodes of color* c_i *needs to be between* $1:q_i$ *and* $1:p_i$*, there exists an algorithm which gives a clustering where* $\forall 1 < i \leq |\mathcal{C}|$*, the ratio of number of nodes of color* c_1 *to color* c_i *in each cluster is between* $1:q_i$ *and* $1:p_i$*, and the total number of disagreements is at most* $\mathcal{O}((\max_{i=1}^{|\mathcal{C}|}\{q_i\})^2) \cdot OPT$*.*

In Section 3.6, we perform an extensive evaluation on real world datasets to demonstrate the unfair results generated by the classical correlation clustering algorithm and evaluate the ability of our algorithm to generate fair clusters without much loss of solution quality.

3.2 Related Work

Introduced by Bansal, Blum and Chawla in 2004 [11], correlation clustering has received tremendous attention in the past decade. The problem is NP-complete, and a series of follow-up work have resulted in better approximation ratio, generalization to weighted graphs etc. [52–54]. This problem captures a wide range of applications including clustering gene expression patterns [55, 56], and the aggregation of inconsistent information [57].

The research in fairness in machine learning has focused on two main directions, coming up with proper notions of fairness and designing fair algorithms. The first direction includes results on statistical parity [58], disparate impact [59], and individual fairness [60]. Second direction includes a bulk of work including fair rankings [61], fair clusterings [7,9,47,50,62], fair voting [63], and fair optimization with matroid constraints [64].

Puleo and Milencovic [65] studied a new version of correlation clustering, where the objective was to make sure the maximum number of disagreements on each vertex is minimized. Their motivation was to make sure individuals are treated fairly. For the case of complete graphs, the result was first improved by Charikar et al. [19] to 7-approximation, and it was further improved by Kalhan et al. [20] to 5-approximation. The best known

bound for the case of general graphs is $\mathcal{O}(\sqrt{n})$ approximation [19]. In a subsequent work, Ahmadi et al. [66] studied the local correlation clustering problem where the objective was to make sure the maximum number of disagreements on each cluster is minimized, and the communities are treated fairly. They achieved an $\mathcal{O}(\sqrt{\log n \cdot \max\{\log(|E^-|), \log(k)\}})$ approximation for general weighted graphs, where $|E^-|$ denotes the number of negative edges and k is the number of clusters in the optimum solution. Their result was improved by Kalhan et al. [20] to $(2+\epsilon)$-approximation.

Chierichetti et al. [7] extended the notion of disparate impact to k-center and k-median, and studied these problems for the case of two groups. Their result was later generalized to multiple groups by Rösner and Schmidt [62]. In this work, we generalize the notion of disparate impact to correlation clustering for multiple colors, and our goal is to make sure the distribution of colors in each cluster is identical to the global distribution. Next, we extend the model introduced by Ahmadian et al. [50] on k-center to correlation clustering to show no color is overrepresented or underrepresented in each cluster.

3.3 Preliminaries

In the correlation clustering problem, an input graph $G = (V, E)$ is given where each edge is labeled positive or negative. The goal is to obtain a clustering that minimizes the total number of disagreements, defined as the number of negative edges trapped inside a cluster plus positive edges that are cut between clusters.

Inspired by the recent developments on fairness in clustering [7, 9, 47], we define a fair variant of correlation clustering problem. In fair correlation clustering problem, given an input graph $G = (V, E)$ each node has a color from set of colors $\{c_1, \cdots, c_{|\mathcal{C}|}\}$. The desired ratio of the number of nodes from color c_1 to color c_i is $1 : p_i, \forall 1 \leq i \leq |\mathcal{C}|$ where $\forall 1 \leq i \leq |\mathcal{C}| : p_i \in \mathcal{Z}^{\geq 1}$. The goal is to find a clustering which guarantees the desired ratios in each cluster while minimizing the total number of disagreements. First, we study fair correlation clustering problem for 2 colors and then extend it to an arbitrary number of colors.

In Section 3.5, we consider the problem of given an instance of correlation clustering where nodes are either red or blue and the ratio of the number of red nodes to blue nodes is between $1 : q, 1 : p$, where $p, q \in \mathcal{Z}^{\geq 1}, p \leq q$. The goal is to form a clustering where the ratio of the number of red nodes to blue

Algorithm 2: Fair Correlation Clustering

Input : $G = (V = V_B \cup V_R, E = E^+ \cup E^-)$
Output: clustering $\mathcal{C}'$
$E' \leftarrow \phi$;
for $u \in V_B$ **do**
  for $v \in V_R$ **do**
    $E' \leftarrow E' \cup (u, v)$;
    $w(u, v) \leftarrow \sum_{w \in V \setminus \{u,v\}} \mathbb{1}_{(u,w) \in E^+, (v,w) \in E^-} + \mathbb{1}_{(u,w) \in E^-, (v,w) \in E^+}$;
$\mathcal{C}' \leftarrow ClassicCorrClust(V_R, E \cap V_R \times V_R)$;
$M \leftarrow$ `min-weight-matching`(V, E', w);
$\forall v \in V_B$, assign v to same cluster as $M(v)$;
return $\mathcal{C}'$;

nodes in each cluster is between $1 : q$ and $1 : p$. Throughout the paper, we use OPT interchangeably for the optimum solution and minimum number of disagreements.

3.4 Warmup: 2 Colors with Ratio 1:1

The goal is to find a clustering which minimizes the total number of disagreements, and the number of red and blue vertices in each cluster are equal. We show a constant approximation algorithm for this problem.

Algorithm 2 presents our approach to generate clusters that obey the fairness constraint while minimizing the total disagreements. First, a weighted bipartite graph from V_B to V_R is constructed (lines 2-8) in the following way: consider a pair of vertices (x, y) where $x \in V_B$ and $y \in V_R$, weight of this edge

$w(x, y)$ is initially set to zero. If (x, y) is a negative edge, increase $w(x, y)$ by 1. For each vertex $z \in V \setminus \{x, y\}$, if the labels of edges (z, x) and (z, y) are different, increase $w(x, y)$ by 1 (line 6). In this way, $w(x, y)$ shows how much the total disagreement increases if x and y are clustered together. Run an α-approximation algorithm for minimizing disagreements on V_R with no fairness constraints (line 9). Since $V_R \subset G$ and there are no fairness constraints on V_R, $OPT_{V_R} \leq OPT_G$. Next, find a minimum weighted matching M from V_B to V_R (line 10). In the end assign each blue vertex to the same cluster as its matched red node (line 11), and return the new clustering. Let $w(M)$ denote weight of this matching. First, we show the following lemma holds:

Lemma 6. $w(M) \leq 2 \cdot OPT_G$.

Proof. Consider the optimal solution and construct an arbitrary matching where both endpoints of each matched edge are in the same cluster. Call this matching M' and assign weights to each matched edge as described in Algorithm 2. First, we show $w(M') \leq 2 \cdot OPT_G$. Consider an edge (v_i, v_j), if they are matched by the M', M' and OPT are paying the same cost for this edge, since the matching is paying for this edge if and only if it is a negative edge trapped inside a cluster, in this case OPT is also paying for it. Assume the case where v_i, v_j are not matched in M'. Let's assume v_i is matched to $v_{i'}$, and v_j is matched to $v_{j'}$. The matching M' could pay for the edge (v_i, v_j) at most

twice, once if the edges (v_i, v_j) and $(v_{i'}, v_j)$ have disagreeing labels, and once if (v_i, v_j) and $(v_i, v_{j'})$ have disagreeing labels. Therefore, $w(M') \leq 2 \cdot OPT$. Therefore, $w(M) \leq w(M') \leq 2 \cdot OPT$ since we are finding a min cost matching M. □

In the following we show this algorithm gives a $3\alpha + 4$-approximation and prove Theorem 5.

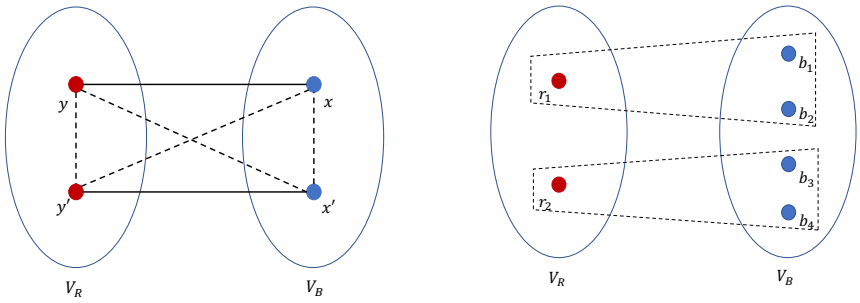
Theorem 5 (Warm-up)**.** *Given a complete unweighted graph $G = (V, E)$ where edges are labeled positive or negative, and the nodes are either red or blue, with an equal number of red and blue nodes, there exists an algorithm which gives a clustering with equal number of red and blue nodes in each cluster and at most $(3\alpha + 4) \cdot OPT$ disagreements, where α is the best approximation ratio for correlation clustering on a complete unweighted graph with minimizing disagreements objective.*[2]

Proof. Consider vertices $x, x' \in V_B$ and $y, y' \in V_R$, where $(x, y) \in M$ and $(x', y') \in M$ (Figure 3.1(a)). In the following, we show that we can pay for all the disagreements within our $(3\alpha + 4) \cdot OPT$ budget.

Case 1: If a disagreement happens on a matched edge (x, y), meaning (x, y) is a negative edge, it is counted in $w(M)$.

Case 2: If a disagreement is on (x', y), two cases may arise:

[2]The best known value of $\alpha = 2.06$, giving 10.18-approximation of our algorithm.



(a) Warm-up case (b) General case

Figure 3.1: Example set of matched nodes for our algorithm.

- Case 2.1: If (x', y) and (y', y) have the same label, since x' and y' are in the same cluster, having a disagreement on (x', y) implies having a disagreement on (y', y). Therefore, if we double the budget needed to pay for the mistakes in V_R, we can also pay for the mistakes of this type.
- Case 2.2: If (x', y) and (y', y) do not have the same label, we are making exactly one mistake on these two edges and the min cost matching M is paying for it.

Case 3: If a disagreement occurs on (x, x'), the following cases might happen:

- Case 3.1: If (x, x'), (x', y) have different labels, we are making exactly one mistake on these two edges and the min cost matching M is paying for it.
- Case 3.2: If (x', x) and (x', y) have the same labels, two cases might

happen:

- Case 3.2.1: (x', y) and (y, y') have the same labels. In this case, (x', x) and (y', y) have the same labels, and x', y' are in the same cluster, also x, y are in the same cluster. Therefore, there is a disagreement on (x', x) if and only if there is a disagreement on (y, y'). By adding another $\alpha \cdot OPT$ to the budget, we can pay for these types of disagreements.
- Case 3.2.2: (x', y) and (y', y) have different labels. In this case, exactly one mistake occurred on (x', y) and (y, y') and matching was paying for it. If no mistakes occur on (x', y), there will be no mistake on (x, x') as well. If a disagreement happens on (x', y), then a disagreement occurs on (x', x). Since M is paying for the disagreement occurred on (x', y), doubling the cost of M in the budget pays for the mistake on (x, x') as well.

At the end, we get a $3\alpha + 4$-approximation algorithm, which complete proof of Theorem 5. □

3.5 Generalization

In this section, we generalize the previously considered model to allow the ratio of colors to be $1 : p$ where $p \in \mathcal{Z}^{\geq 1}$ in Section 3.5 and allow more than 2 colors in Section 3.5.

Two Colors with Ratio 1:p

The algorithm for this case is similar to Algorithm 2 with some minor differences; the matching M constructed from V_R to V_B is a b-matching where the degrees of vertices in V_R are p, and the degrees of vertices in V_B are 1. Let $w(M)$ denote weight of this matching. Next, run an α-approximation correlation clustering on a subset of G which includes the red vertex from each hyper-node (i.e. a collection of matched nodes). Theorem 6 shows a guarantee for this algorithm. Before proving Theorem 6, we need to show the following lemma holds:

Lemma 7. $w(M) \leq 2p \cdot OPT$.

Proof. Consider the OPT solution, and construct an arbitrary b-matching M' from red nodes to blue nodes where degree of each red node is p, and degree of each blue node is 1, and for each matched edge both its endpoints belong to

the same cluster. Call this matching M'. First, we show $w(M') \leq 2p \cdot OPT_G$. Consider an edge between arbitrary vertices v_i and v_j, such that they are not matched in M'. If a disagreement occurs on the edge between (v_i, v_j) in OPT_G, this disagreement could have been counted at most $2p$ times in $w(M')$. Therefore $w(M') \leq 2p \cdot OPT_G$. Since M is a min cost b-matching satisfying degree constraints: $w(M) \leq w(M') \leq 2p \cdot OPT_G$ □

Now we are ready to prove Theorem 6.

Theorem 6 (Two Colors)**.** *Given a complete unweighted graph $G = (V, E)$ where edges are labeled positive or negative, and the nodes are either red or blue where the ratio of number of red nodes to the number of blue nodes is $1 : p$ for $p \in \mathcal{Z}^{\geq 1}$, there exists an algorithm which gives a clustering with ratio $1 : p$ of number of red to blue nodes in each cluster and at most $\mathcal{O}(p^2) \cdot OPT$ disagreements.*

Proof. In the following, we show we can pay for all the disagreements within a $\left((p^2 + 2p) \cdot \alpha + 4p^2\right) \cdot OPT$ budget.

Case 1: In Figure 3.1(b), consider a disagreement between a red vertex (r_1) and a blue (b_3) node from different hyper-nodes. Two cases might happen:

- Case 1.1: If edges (r_1, b_3) and (r_1, r_2) have disagreeing labels, then cost of the edge (r_2, b_3) counted in the matching is paying for it.

- Case 1.2: If edges (r_1, b_3) and (r_1, r_2) have the same signs, the disagreement on (r_1, b_3) could be charged to the edge (r_1, r_2). The number of such edges charged to (r_1, r_2) is at most $2p$.

Case 2: There exists disagreement between two blue nodes from different hyper-nodes, like (b_1, b_3) in Figure 3.1(b).

- Case 2.1: Edges (b_1, b_3) and (r_1, b_3) are disagreeing. Then the cost of edge (r_1, b_1) included in the cost of the matching is paying for it.
- Case 2.2: Edges (b_1, b_3) and (r_1, b_3) have the same labels and have different labels with (r_1, r_2). We charge the disagreement on (b_1, b_3) to the edge (r_2, b_3). There are p choices for b_1, therefore at most p edges of this type, plus the edge (r_1, b_3) are charged to the edge (r_2, b_3), whereas M is paying 1 for the disagreement between (r_1, r_2) and (r_1, b_3). Therefore, we need to account for $p + 1$ times the matching cost to account for all edges of this type.
- Case 2.3: Edges $(b_1, b_3), (r_1, b_3), (r_1, r_2)$ all have the same labels. There are p^2 choices for a pair of blue nodes like (b_1, b_3), and disagreements on these edges could be charged to (r_1, r_2).

Case 3: A disagreement between two blue nodes in the same hyper-node, b_1 and b_2 which means (b_1, b_2) is a negative edge. If (r_1, b_1) is positive then (r_1, b_2)'s contribution in the matching cost captures it. Similarly, if (r_1, b_2) is a

Algorithm 3: Fair Correlation Clustering for Multiple Colors

Input : $G = (V = V_{c_1} \cup V_{c_2} \cdots \cup V_{c_\mathcal{C}}, E = E^+ \cup E^-)$
Output: clustering $\mathcal{C}'$
for $1 < i \leq |\mathcal{C}|$ **do**
  $E'_i \leftarrow \phi$;
  for $u \in V_{c_i}$ **do**
    for $v \in V_{c_1}$ **do**
      $E'_i \leftarrow E'_i \cup (u, v)$;
      $w(u, v) \leftarrow \sum_{w \in V_i \setminus \{u,v\}} \mathbb{1}_{(u,w) \in E^+, (v,w) \in E^-} + \mathbb{1}_{(u,w) \in E^-, (v,w) \in E^+}$;
  $M_i \leftarrow$ `min-weight-b-matching`$(V_1 \cup V_{c_i}, E'_i, w)$
$\mathcal{C}' \leftarrow ClassicCorrClust(V_{c_1}, E \cap V_{c_1} \times V_{c_1})$;
for $1 < i \leq |\mathcal{C}|$ **do**
  $\forall v \in V_{c_i}$, assign v to the same cluster as $M_i(v)$;
return $\mathcal{C}'$;

positive edge then the (r_1, b_2)'s contribution in matching cost captures this. If both (r_1, b_1) and (r_1, b_2) are negative edges then we can charge both the edges 1/2. The total number of times an edge (r_1, b_1) is charged is at most $p - 1$ as there can be a maximum of $p - 1$ negative edges from b_1.

There is a total of $p^2 + 2p$ charges on edges between red nodes (Cases 1.2 and 2.3) accounting for the total cost to be $(p^2 + 2p)C$, where C is the correlation clustering objective on red vertices. Similarly, we charge each matched edge at most $p+1$ times their weight in Case 2.2 and at most $p-1$ times their weight in Case 3, the total contribution to the final objective is $2p \cdot w(M)$. All charges required to handle cases 1.1 and 2.1 do not add any additional cost to the objective as they are already accounted for edges considered in $2p \cdot w(M)$. Hence, by applying Lemma 7, the total objective value of returned clusters is

at most:

$$(p^2+2p)C + (p+1)\cdot w(M) \leq \Big((p^2+2p)\cdot\alpha + 2p\times 2p\Big)\cdot OPT$$

Therefore the approximation ratio is $\mathcal{O}(p^2)$, and this completes proof of Theorem 6. □

Multiple Colors

Our results could be extended to the case of multiple colors. Assume there are $\mathcal{C}$ colors $\{c_1, c_2, \cdots, c_{|\mathcal{C}|}\}$, and the ratio of number of nodes of color c_1 to color c_i is $1 : p_i$ $(\forall 1 < i \leq |\mathcal{C}|)$, where $p_i \in \mathcal{Z}^{\geq 1}$. In this case, we get an approximation ratio of $\mathcal{O}((\max_{i=1}^{|\mathcal{C}|}\{p_i\})^2 \cdot |\mathcal{C}|^2)$. Algorithm 3 is a generalization of Algorithm 2. In this algorithm a set of b-matchings $\{M_i : 1 < i \leq |\mathcal{C}|\}$ are constructed. Each matching M_i is between nodes of color c_1 and c_i and degree of each node of color c_1 is p_i, and degree of each node of color c_i is 1. Analysis of this algorithm is similar to the analysis of the algorithm for 2 colors. First, we need to show the following lemma holds, which can be proved similar to the way Lemma 7 was proved.

Lemma 8. *In each matching M_i constructed in Algorithm 3, $w(M_i) \leq 2p_i \cdot OPT$.*

Now we are ready to prove the following theorem holds:

Theorem 7 (Multiple Colors)**.** *Given a complete unweighted graph $G = (V, E)$ where edges are labeled positive or negative, and each node has exactly one of the colors $\{c_1, \cdots, c_{|\mathcal{C}|}\}$, and the ratio of the number of nodes of color c_1 to color c_i is $1 : p_i$ $(\forall 1 < i \leq |\mathcal{C}|)$, where $p_i \in \mathcal{Z}^{\geq 1}$, there exists an algorithm where the distribution of colors in each cluster is the same as the global distribution, and the total number of disagreements is at most $\mathcal{O}((\max_{i=1}^{|\mathcal{C}|}\{p_i\})^2 \cdot |\mathcal{C}|^2) \cdot OPT$.*

Proof. Let $p_{\max} = \max_{i=1}^{|\mathcal{C}|}\{p_i\}$. In the following we show how to pay for all the disagreements within a $\mathcal{O}((p_{\max})^2 \cdot |\mathcal{C}|^2) \cdot OPT$ budget. For simplicity we assume color c_1 is red, and there are at least two other colors blue (c_2) and green (c_3). Consider the following cases:

Case 1: This case is similar to Case 1 in Section 3.5. Consider a disagreement between a red vertex (let's say r_1), and a node of a different color (let's say blue node b_3) such that r_1 and b_3 are not matched by matching M_2. Let's assume M_2 matches b_3 to r_2.

- Case 1.1: If edges (r_1, b_3) and (r_1, r_2) have disagreeing labels, then cost of the edge (r_2, b_3) counted in the $w(M_2)$ is paying for it.
- Case 1.2: If edges (r_1, b_3) and (r_1, r_2) have the same signs, the disagreement on (r_1, b_3) could be charged to the edge (r_1, r_2). The number of such edges charged to (r_1, r_2) is $2p_2$ (and $2p_i$ in general if instead of b_3

we considered a node of color c_i).

Case 2: There exists a disagreement between two non-red nodes from two different hyper nodes, let's say between nodes b_1, g_1. Let's assume b_1 is matched to r_1 by M_2 (the matching between red and blue nodes), and g_1 is matched to r_2 by M_3 (the matching between red and green nodes).

- Case 2.1: Edges (b_1, g_1) and (r_1, g_1) are disagreeing. Then the cost of edge (r_1, b_1) included in the cost of $w(M_2)$ is paying for it.
- Case 2.2: Edges (b_1, g_1) and (r_1, g_1) have the same labels and have different labels with (r_1, r_2). We charge the disagreement on (b_1, g_1) and (r_1, g_1) to the edge (r_2, g_1). There are $(|\mathcal{C}| - 1) \cdot p_{\max}$ choices for b_1 which are all the nodes that are matched to r_1 in all the matchings $M_2, \cdots, M_{|\mathcal{C}|}$. Therefore in this case, at most $(|\mathcal{C}| - 1) \cdot p_{\max} + 1$ edges, are charged to the edge (r_2, g_1), when the matching edge between (r_2, g_1) is paying 1 for the disagreement between (r_1, r_2) and (r_1, g_1).
- Case 2.3: Edges $(b_1, g_1), (r_1, g_1), (r_1, r_2)$ all have the same labels. We charge disagreements on these edges to (r_1, r_2). There are at most $((|\mathcal{C}| - 1) \cdot p_{\max})^2$ choices for a pair of non-red nodes like (b_1, g_1), charged to (r_1, r_2).

Case 3: This case captures the disagreement between two non-red nodes in the same hyper-node and is similar to Case 3 in Section 3.5.

There is a total of $((|\mathcal{C}|-1)\cdot p_{\max})^2 + 2p_{\max}$ charges on edges between red nodes (Cases 1.2 and 2.3) accounting for the total cost to be $(((|\mathcal{C}|-1)\cdot p_{\max})^2+2p_{\max})C$, where C is the correlation clustering objective on red vertices. Similarly, we charge each matched edge $|\mathcal{C}|\cdot p_{\max}+1$ times (Case 2.2) and $p_{\max}-1$ times (Case 3), thereby contributing $\sum_{i=2}^{|\mathcal{C}|}((|\mathcal{C}|+1)\cdot p_{\ max})\cdot w(M_i)$ to the final objective. Considering Lemma 8, we can conclude the approximation ratio is $\mathcal{O}((p_{\max})^2\cdot|\mathcal{C}|^2)$, and this completes the proof. □

Note: When $p_{max} = 1$, we can perform classical correlation clustering on nodes of any color $C_i \in \mathcal{C}$ and picking the one that has minimum value. This optimization helps reduce the quadratic dependence of total disagreements of case 2.3 on $|\mathcal{C}|$ to linear dependence.

Avoiding Over-representation

In this section, we consider the model defined by Ahmadian et al. [50] for k-center problem. Their goal is to make sure given a parameter $0 \leq \alpha \leq 1$, maximum fraction of nodes in a cluster having a specific color is at most α times the size of the cluster. We consider the following problem: consider two colors red and blue. Our goal is to make sure the ratio of the number of red nodes to the number of blue nodes in each cluster is between $1:q$ and $1:p$

where $1 \leq p \leq q$ and $p, q \in \mathcal{Z}^{\geq 1}$. The algorithm discussed in Section 3.5 could be modified to handle this variation of the problem in the following way: when finding a minimum cost b-matching M, put the degree constraint on each red node to be between p and q, and let the degree constraint on each blue node to be 1. Therefore in the minimum cost matching M, each connected component has 1 red node and at least p and at most q blue nodes. Next, run an α-approximation correlation clustering on a subset of G which includes the red vertex from each hyper-node (i.e. a collection of matched nodes). The rest of the algorithm is similar to the algorithm discussed in Section 3.5. Using a similar analysis, we show the approximation ratio is $\mathcal{O}(q^2)$. First, we need to show the following lemma holds:

Lemma 9. $w(M) \leq 2q \cdot OPT$.

Proof. Given the optimum solution OPT, we can show a b-matching M' could be constructed where endpoints of each matched edge belong to the same cluster, and the degree of each red node is at least p and at most q. In the OPT solution, in each cluster, the ratio of the number of red to blue nodes is between $1 : q$ and $1 : p$. Consider a specific cluster $\mathcal{X}$ in the OPT solution, let n_r, n_b denote the number of red and blue nodes in this cluster respectively. Therefore, $n_r \cdot p \leq n_b \leq n_r \cdot q$. Construct a b-matching inside $\mathcal{X}$ as following: first assign p distinct blue nodes to each red node in $\mathcal{X}$. If any blue nodes in

$\mathcal{X}$ are left un-assigned, assign them to any red node in $\mathcal{X}$ which is assigned to less than q blue nodes. Since $n_b \leq q \cdot n_r$, we can find a b-matching with desired properties in each cluster of the OPT solution. M' is the union of the b-matchings formed in all the clusters. First, we show $w(M') \leq 2q \cdot OPT_G$. Consider an edge between arbitrary vertices v_i and v_j, such that they are not matched in M'. If a disagreement occurs on the edge between (v_i, v_j) in OPT_G, this disagreement could have been counted at most $2q$ times in $w(M')$. Therefore $w(M') \leq 2q \cdot OPT_G$. Since M is a min cost b-matching satisfying degree constraints:

$$w(M) \leq w(M') \leq 2q \cdot OPT_G$$

□

Now we are ready to show the following theorem holds:

Theorem 8 (Avoiding Over-Representation)**.** *Given a complete unweighted graph $G = (V, E)$ where edges are labeled positive or negative, and nodes are colored red or blue, and two ratios $1 : p, 1 : q$ where $p, q \in \mathcal{Z}^{\geq 1}, p \leq q$, where ratio of the total number of red nodes to the total number of blue nodes is between $1 : q$ and $1 : p$, there exists an algorithm which gives a clustering where the ratio of number of red nodes to blue nodes in each cluster is between $1 : q$ and $1 : p$, and the total number of disagreements is at most $\mathcal{O}(q^2) \cdot OPT$.*

Proof. In the following, we show we can pay for all the disagreements within a $\left((q^2+2q)\cdot\alpha+4q^2\right)\cdot OPT$ budget.

Case 1: In Figure 3.1(b), consider a disagreement between a red vertex (r_1) and a blue (b_3) node from different hyper-nodes. Two cases might happen:

- Case 1.1: If edges (r_1, b_3) and (r_1, r_2) have disagreeing labels, then cost of the edge (r_2, b_3) counted in the matching is paying for it.
- Case 1.2: If edges (r_1, b_3) and (r_1, r_2) have the same signs, the disagreement on (r_1, b_3) could be charged to the edge (r_1, r_2). The number of such edges charged to (r_1, r_2) is at most $2q$.

Case 2: There exists a disagreement between two blue nodes from two different hyper-nodes, like (b_1, b_3) in Figure 3.1(b).

- Case 2.1: Edges (b_1, b_3) and (r_1, b_3) are disagreeing. Then the cost of edge (r_1, b_1) included in the cost of the matching is paying for it.
- Case 2.2: Edges (b_1, b_3) and (r_1, b_3) have the same labels and have different labels with (r_1, r_2). We charge the disagreement on (b_1, b_3) to the edge (r_2, b_3). There are p choices for b_1, therefore at most p edges of this type, plus the edge (r_1, b_3) are charged to the edge (r_2, b_3), when M is paying 1 for the disagreement between (r_1, r_2) and (r_1, b_3). Therefore, we need to account for at most $q+1$ times the matching cost to account for all edges of this type.

- Case 2.3: Edges $(b_1, b_3), (r_1, b_3), (r_1, r_2)$ all have the same labels. There are at most q^2 choices for a pair of blue nodes like (b_1, b_3), and disagreements on these edges could be charged to (r_1, r_2).

Case 3: A disagreement between two blue nodes in the same hyper-node, b_1 and b_2 which means (b_1, b_2) is a negative edge. If (r_1, b_1) is positive then (r_1, b_2)'s contribution in the matching cost captures it. Similarly, if (r_1, b_2) is a positive edge then the (r_1, b_2)'s contribution in matching cost captures this. If both (r_1, b_1) and (r_1, b_2) are negative edges then we can charge both the edges $1/2$. The total number of times an edge (r_1, b_1) is charged is at most $q - 1$ as there can be a maximum of $q - 1$ negative edges from b_1.

There is a total of $q^2 + 2q$ charges on edges between red nodes (Cases 1.2 and 2.3) accounting for the total cost to be $(q^2 + 2q)C$, where C is the correlation clustering objective on red vertices. Similarly, we charge each matched edge at most $q + 1$ times their weight in Case 2.2 and at most $q - 1$ times their weight in Case 3, the total contribution to the final objective is at most $(2q) \cdot w(M)$. All the charges required to handle cases 1.1 and 2.1 do not add any additional cost to the objective as they are already accounted for the edges considered in $(2q) \cdot w(M)$. Hence, by applying Lemma 9, the total objective

value of returned clusters is at most:

$$(q^2 + 2q)C + (q + 1) \cdot w(M) \leq \Big((q^2 + 2q) \cdot \alpha + 2q \times 2q\Big) \cdot OPT$$

Therefore the approximation ratio is $\mathcal{O}(q^2)$, and this completes proof of Theorem 8. □

In the case of multiple colors, where the goal is that in each cluster the ratio of the number of nodes of color c_1 to color c_i be between $1 : p_i$ and $1 : q_i$ where $\forall 1 < i \leq |\mathcal{C}|, p_i \leq q_i$ and $p_i, q_i \in \mathcal{Z}^{\geq 1}$, Algorithm 3 could be modified to handle this case: in each iteration, find a minimum cost weighted b-matching M_i where the degree of each node of color c_1 is between p_i and q_i, and the degree of each node of color c_i is 1. By applying Lemma 9 to each matching M_i, one can see $w(M_i) \leq 2q_i \cdot OPT$. Next, we run an α-approximation on the nodes of color c_1, and for each fixed vertex u of color c_1, all the vertices that are matched to u using any of the matchings $\{M_2, \cdots, M_{|\mathcal{C}|}\}$ go to the same cluster as u. We show the following theorem holds for this scenario:

Theorem 9 (Avoiding Over-Representation with Multiple Colors)**.** *Given a complete unweighted graph $G = (V, E)$ where edges are labeled positive or negative, and each node has exactly one of the colors $\{c_1, \cdots, c_{\mathcal{C}}\}$, and two ratios $1 : p_i, 1 : q_i$ for each color c_i where $p_i, q_i \in \mathcal{Z}^{\geq 1}, p_i \leq q_i$, where ratio*

of the total number of nodes of color c_1 to the total number of nodes of color c_i needs to be between $1 : q_i$ and $1 : p_i$, there exists an algorithm which gives a clustering where $\forall 1 < i \leq |\mathcal{C}|$, the ratio of number of nodes of color c_1 to color c_i in each cluster is between $1 : q_i$ and $1 : p_i$, and the total number of disagreements is at most $\mathcal{O}((\max_{i=1}^{|\mathcal{C}|}\{q_i\})^2) \cdot OPT$.

Proof. Let $q_{max} = \max\{q_2, \cdots, q_{|\mathcal{C}|}\}$. In the following we show how to pay for all disagreements within a $\mathcal{O}((q_{\max}^2) \cdot |\mathcal{C}|^2) \cdot OPT$ budget. For simplicity let's assume color c_1 is red, and there are at least two other colors blue (c_2) and green (c_3). Consider the following cases:

Case 1: Consider a disagreement between a red vertex (let's say r_1), and a node of a different color (let's say blue node b_3) such that r_1 and b_3 are not matched by matching M_2. Let's assume M_2 matches b_3 to r_2.

- Case 1.1: If edges (r_1, b_3) and (r_1, r_2) have disagreeing labels, then cost of the edge (r_2, b_3) counted in the $w(M_2)$ is paying for it.
- Case 1.2: If edges (r_1, b_3) and (r_1, r_2) have the same signs, the disagreement on (r_1, b_3) could be charged to the edge (r_1, r_2). The number of such edges charged to (r_1, r_2) is at most $2q_2$ (and $2q_i$ in general if instead of b_3 we considered a node of color c_i).

Case 2: There exists a disagreement between two non-red nodes from two different hyper nodes, let's say between nodes b_1, g_1. Let's assume b_1 is matched

to r_1 by M_2 (the matching between red and blue nodes), and g_1 is matched to r_2 by M_3 (the matching between red and green nodes).

- Case 2.1: Edges (b_1, g_1) and (r_1, g_1) are disagreeing. Then the cost of edge (r_1, b_1) included in the cost of $w(M_2)$ is paying for it.
- Case 2.2: Edges (b_1, g_1) and (r_1, g_1) have the same labels and have different labels with (r_1, r_2). We charge the disagreement on (b_1, g_1) and (r_1, g_1) to the edge (r_2, g_1). There are $(|\mathcal{C}| - 1) \cdot q_{\max}$ choices for b_1 which are all the nodes that are matched to r_1 in all the matchings $M_2, \cdots, M_{|\mathcal{C}|}$. Therefore in this case, at most $(|\mathcal{C}| - 1) \cdot q_{\max} + 1$ edges, are charged to the edge (r_2, g_1), when the matching edge between (r_2, g_1) is paying 1 for the disagreement between (r_1, r_2) and (r_1, g_1).
- Case 2.3: Edges $(b_1, g_1), (r_1, g_1), (r_1, r_2)$ all have the same labels. We charge disagreements on these edges to (r_1, r_2). There are at most $((|\mathcal{C}| - 1) \cdot q_{\max})^2$ choices for a pair of non-red nodes like (b_1, g_1), charged to (r_1, r_2).

Case 3: This case captures the disagreement between two non-red nodes in the same hyper-node and is similar to Case 3 in Section 3.5.

There is a total of $((|\mathcal{C}| - 1) \cdot q_{\max})^2 + 2q_{\max}$ charges on edges between red nodes (Cases 1.2 and 2.3) accounting for the total cost to be $(((|\mathcal{C}| - 1) \cdot q_{\max})^2 + 2q_{\max})C$, where C is the correlation clustering objective on red vertices.

Similarly, we charge each matched edge $|\mathcal{C}| \cdot q_{\max} + 1$ times (Case 2.2) and $q_{\max}-1$ times (Case 3), thereby contributing $\sum_{i=2}^{|\mathcal{C}|}((|\mathcal{C}|+1)\cdot q_{\ max})\cdot w(M_i)$ to the final objective. Since $w(M_i) \leq 2q_i \cdot OPT$, we can conclude the approximation ratio is $\mathcal{O}((q_{\max})^2 \cdot |\mathcal{C}|^2)$, and this completes proof of Theorem 7. □

Hardness

Consider a complete graph $G = (V, E = E^+ \cup E^-)$. Consider a new complete graph H of $2|V|$ nodes which is constructed by duplicating the nodes of G (say V and $V' = \{u' \mid u \in V\}$). We can assume the nodes of V to be colored red and V' can be considered as the mirror image of V colored blue. Each pair of nodes $u, v \in V$ are connected in the same way as E. A positive edge is added between u and u' for all $u \in V$, where u' is the mirror image of u. For all $u \in V, v' \in V'$, the edge between (u, v') has the same label as (u, v) where v is the mirror of v'. The graph H restricted to vertices V' is referred to as $G' = (V', E')$ (as shown in Figure 3.2).

Consider a clustering of H with equal number of red and blue vertices in each cluster (say $\mathcal{C}'$). Now, we calculate the disagreements on the edges between nodes of V and V' to bound the total disagreements of $\mathcal{C}'$. A disagreement edge $(u', v') \in E'$ leads to the following scenarios.

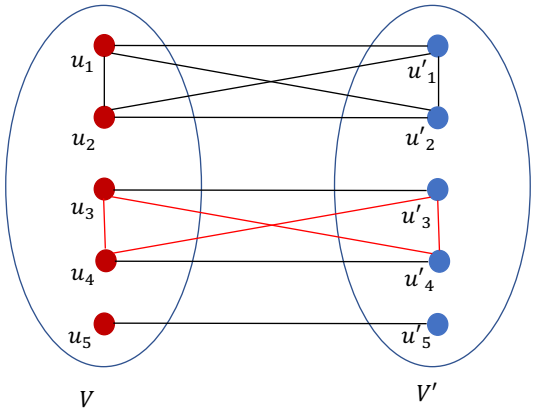


Figure 3.2: Red edges have negative labels, and black edges have positive labels.

- Case 1: If $(u', v') \in E^-$: Therefore nodes u' and v' belong to same cluster.
  - Case 1.1: If u, v belong to the same cluster as u' and v', then edges (u, v') and (u', v) are mistakes.
  - Case 1.2: u belongs to same cluster as u' and v' but v belongs to a different cluster. Edges (v, v') and (u, v') are the mistakes.
  - Case 1.3: u and v belong to different cluster from u' and v', edges (u, u') and (v, v') are mistakes
- Case 2: If (u', v') is a positive edge. This means u' and v' belong to different clusters.
  - Case 2.1: If u belongs to same cluster as u' and v belongs to same cluster as v' then edges (u, v') and (u', v) are the mistakes.
  - Case 2.2: If u belongs to different cluster from u' and v belongs to

different cluster from v' then (u, u') and (v, v') are the mistakes.

- Case 2.3: If u belongs to different cluster from u' and v, v' belong to the same cluster:
  - Case 2.3.1: If u and v' belong to different clusters then (u, v') and (u, u') are mistakes.
  - Case 2.3.2: If u and v' belong to the same cluster then (u, u') and (u', v) are mistakes.

This shows that for every disagreement $(u', v') \in E'$, there exist at least 2 disagreements in the edges between $\{u', v'\}$ and $\{u, v\}$: $(u, v'), (u', v), (u, u')$ and (v, v'). If the disagreements on the subgraph of H limited to (V', E') is $O_{G'}$, then the total disagreements on the edges between V and V' is at least $2O_{G'}$. Hence, the total disagreements of $\mathcal{C}'$ is at least $3O_{G'} + O_G$, where O_G is the disagreements on subgraph limited to $G = (V, E)$. Symmetrically performing the above-mentioned analysis on $(u, v) \in E$, the total disagreements of $\mathcal{C}'$ is at least $3O_G + O_{G'}$. Hence the total number of disagreements of $\mathcal{C}'$ is at least $\max\{3O_{G'} + O_G, 3O_G + O_{G'}\}$, which is minimized when $O_G = O_{G'} = OPT_G$. This is minimized when each red node and its mirror image belong to the same cluster. By discarding the nodes of V' from the optimal solution of H, we get the optimal solution of classical correlation clustering on G.

3.6 Experiments

In this section, we empirically evaluate our algorithm along with some baselines on real world datasets. We show that the clusters generated by classical correlation clustering algorithm are unfair and our algorithm returns fair clusters without much loss in the quality of the clusters[3].

Datasets. We consider the following datasets.

1. *Bank*[4]. This dataset comprises of phone call records of a marketing campaign run by a Portuguese bank. The *marital* status of the clients is considered feature to ensure fairness.
2. *Adult*[5]. Each record in the dataset represents a US citizen whose information was collected during 1994 census. We consider the feature *sex* for fairness.
3. *Medical Expenditure*[6]. The dataset contains medical information of various patients collected for research purposes. The *race* attribute is considered for fairness.
4. *Compas*[7]. The dataset comprises of records of criminal trials used to

[3]We plan to opensource the code repository along with readme files. The code is available at https://www.dropbox.com/sh/x99fz8mdgfyec8v/AAD174qwFGfBSP2AQyPPwC1Fa?dl=0

[4]https://archive.ics.uci.edu/ml/datasets/Bank+Marketing

[5]https://archive.ics.uci.edu/ml/datasets/adult

[6]https://meps.ahrq.gov/mepsweb/

[7]https://github.com/propublica/compas-analysis

analyze criminal recidivism. We consider race attribute for fairness.

Each record in the above-mentioned datasets are considered as the nodes of the graph and the edge sign is determined by attribute similarity between the nodes. We consider a random sample of 1000 nodes in the above datasets for our experiments.

Baselines. We compare the quality of clusters returned by our algorithm with the following baselines focused towards minimizing disagreements and ensure fairness. (i) `CC` – The classical correlation clustering algorithm [52] that guarantees a 3-approximation of the optimal solution but does not ensure fairness (iii) `wMatch` – It generates a matching between nodes of different color as discussed in Algorithm 2 (iv) `uFairCC` – Same as our algorithm with a difference that the matching component considers unit weight on inter-color edge. (v) `CCMerge` – This algorithm runs classical correlation clustering algorithm to generate initial clusters and then greedily add nodes to the clusters in decreasing size, so as to ensure fairness constraints.

All the algorithms were implemented by us in Python using the networkx library on a 64GB RAM server. We run each algorithm 5 times and report average results. We calculate the total disagreements of the returned clusters to evaluate their quality. We denote our algorithm by `FairCC`.

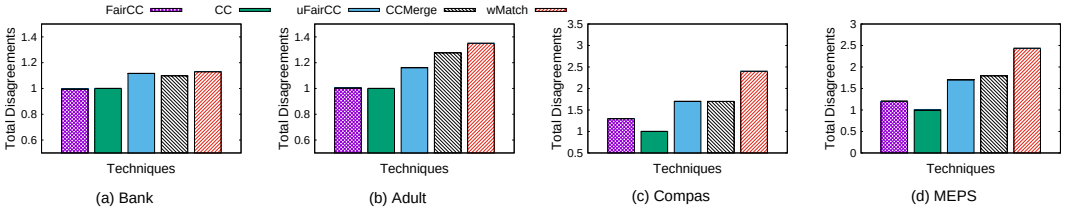


Figure 3.3: Comparison of total disagreements for the different baselines with a constraint of ratio of two colors to be between 1:1 and 1:2.

Solution Quality

This section compares the quality of clusters returned by the different algorithms for the specified distribution of features in each cluster.

Fair proportion. Figure 3.3 compares the total disagreements of the clusters returned by different algorithms. We observe similar trends across all the datasets. The clusters returned by `CC` do not obey the fairness constraint but all the other techniques ensure fairness. Across all datasets, `FairCC` achieves the minimum value of total disagreements as compared to the baselines that ensure fairness. Additionally, the loss in quality of clusters to achieve fairness as compared to `CC` is quite low. The matching returned by `wMatch` is same as that of `FairCC` but it achieves poor quality due to a number of positive edges going across the different components. The `CCMerge` algorithm ends up merging nodes which are connected by negative edges to ensure fairness, thereby losing on quality. `uFairCC` is same as our proposed solution except that the matching component between nodes of different colors does not consider

weights. Superior performance of `FairCC` as compared to `uFairCC` justifies the benefit of our construction of a weighted bipartite graph to match nodes of different colors.

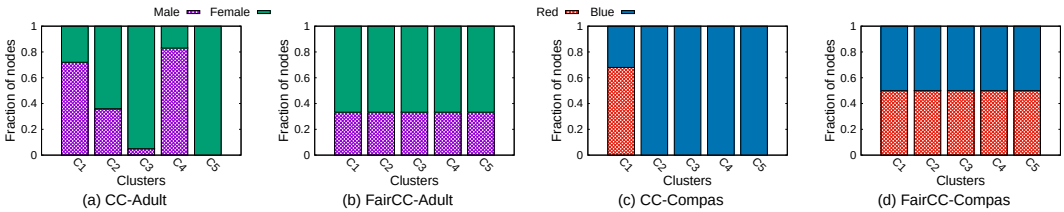


Figure 3.4: Distribution of nodes of different colors in top 5 clusters generated by the algorithms.

Figure 3.4 shows distribution of top-5 clusters generated by `CC` and `FairCC` on Adult and Compas. The skew in distribution of the nodes of two colors in the clusters demonstrates the extent of unfairness in the results generated by classical correlation clustering algorithm. On the other hand, `FairCC` achieves the required fairness constraint in all clusters without losing much in quality. On increasing the range of plausible fraction of two colors, the total disagreements of `FairCC` go down but the trends remain similar.

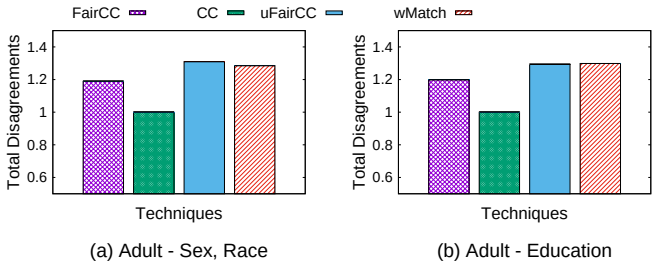


Figure 3.5: Comparison of clusters returned by our algorithm and baselines for instances with more than two colors for Adult dataset. We omit `CCMerge` as it does not generate fair clusters.

Multiple colors. Figure 3.5 compares the performance of `FairCC` with other baselines. Similar to the case of 2 colors, the quality of `FairCC` is not much worse than that of `CC` and is better than any other baseline. This comparison does not plot `CCMerge` as it does not generate clusters that obey fairness.

Running Time. `FairCC` runs in two stages. The first stage identifies a weighted matching between the nodes of different color followed by correlation clustering on one of the colors. On all the datasets, our algorithm ran in less than 10 minutes. For a graph of n nodes, with the increase in number of colors the size of subgraph constructed for matching reduces and total running time does not increase.

3.7 Authors

This Chapter was written by Saba Ahmadi, Sainyam Galhotra, Barna Saha, and Roy Schwartz. It is under submission to the Journal of Machine Learning [10].

4 Min-Max Correlation Clustering via MultiCut

4.1 Introduction

Correlation clustering is a fundamental optimization problem introduced by Bansal, Blum and Chawla [11]. In this problem, we are given a general weighted graph where each edge is labeled positive or negative. The goal is to obtain a partitioning of the vertices into an arbitrary number of clusters that agrees with the edge labels as much as possible. That is, a clustering that minimizes disagreements, which is the weight of positive edges between the clusters plus the weight of negative edges inside the clusters. In addition, correlation clustering captures some fundamental graph cut problems including min s-t cut, multiway cut and multicut. Correlation clustering has been studied extensively for more than a decade [52, 54, 67–69]. Most of the papers

have focused on a global min-sum objective function, i.e. minimizing total number of disagreements or maximizing the total number of agreements.

In recent work, Puleo and Milenkovic [15] introduced a local vertex-wise min-max objective for correlation clustering which bounds the maximum number of disagreements on each node. This problem arises in many community detection applications in machine learning, social sciences, recommender systems and bioinformatics [16–18]. This objective function makes sure each individual has a minimum quality within the clusters. They showed this problem is NP-hard even on un-weighted complete graphs, and developed an $O(1)$ approximation algorithm for unweighted complete graphs. Charikar et al. [19] improved upon the work by Puleo and Milenkovic [15] for complete graphs by giving a 7 approximation. For general weighted graphs, their approximation bound is $O(\sqrt{n})$ where n is the number of vertices. Both these algorithms rely on LP rounding, based on a standard linear program relaxation for the problem. In contrast, for the global minimization objective an $O(\log n)$-approximation can be obtained [69]. Therefore, the local objective for correlation clustering seems significantly harder to approximate than the global objective.

In this work, we propose a new local cluster-wise min-max objective for correlation clustering – minimizing the maximum number of disagreements

of each cluster. This captures the case when we wish to create communities that are harmonious, as global min sum objectives could create an imbalanced community structure. This new local objective guarantees fairness to communities instead of individuals. To name a few applications for this new objective, consider a task of instance segmentation in an image which can be modeled using correlation clustering [70, 71]. A cluster-wise min-max objective makes sure each detected instance has a minimum quality. Another example is in detecting communities in social networks, this objective makes sure there are no communities with lower quality compared to the other communities. No hardness results are known for the cluster-wise min-max objective.

A similar objective was proposed for the multiway cut problem by Svitkina and Tardos [72]. In the min-max multiway cut problem, given a graph G and k terminals the goal is to get a partitioning of G of size k that separates all terminals and the maximum weight of cut edges on each part is minimized. Svitkina and Tardos [72] showed an $\mathcal{O}(\log^3 n)$ approximation algorithm for min-max multiway cut on general graphs (this bound immediately improves to $\mathcal{O}(\log^2 n)$ by using better bisection algorithms). Bansal et al. [73] studied a graph partitioning problem called min-max k-partitioning from a similar perspective. In this problem, given a graph $G = (V, E)$ and $k \geq 2$ the goal is to partition the vertices into k roughly equal parts $S_1, \cdots, S_k$ while minimizing

$\max_i \delta(S_i)$. They showed an $\mathcal{O}(\sqrt{\log n \log k})$ approximation algorithm for this problem. They also improved the approximation ratio given by Svitkina et al. [72] for min-max multiway cut to $\mathcal{O}(\sqrt{\log n \log k})$. Bansal et al.'s seminal work [73] uses the concept of orthogonal separators introduced by Chlamtac et al. [74] to achieve their result.

4.2 Results & High-Level Ideas

In this chapter, we propose an approximation algorithm for the problem of min-max correlation clustering.

Definition 1. *(Min-max Correlation Clustering) Let $G = (V, E)$ be an edge-weighted graph such that each edge is labeled positive or negative. The min-max correlation clustering problem asks for a partitioning of the nodes (a clustering) where the maximum disagreement of a cluster is minimized. Disagreement of a cluster C is the weight of negative edges with both endpoints inside C plus the weight of positive edges with exactly one endpoint in C.*

We prove the following theorem for min-max correlation clustering.

Theorem 1. *Given an edge weighted graph $G = (V, E)$ on n vertices such that each edge is labeled positive or negative, there exists a polynomial time algorithm which outputs a clustering $\mathcal{C} = \{C_1, \cdots, C_{|\mathcal{C}|}\}$ of G such that the*

disagreement on each $C_i \in \mathcal{C}$ is at most $\mathcal{O}(\sqrt{\log n \cdot \max\{\log(|E^-|), \log(k)\}}) \cdot OPT$; where OPT is the maximum disagreement on each cluster in an optimal solution of min-max correlation clustering, k is the number of clusters in the optimum solution, and $|E^-|$ denotes the number of negative edges in G.

In order to prove Theorem 1, we give a reduction from the problem of min-max correlation clustering to a problem which we call min-max multicut.

Definition 2. *(Min-max Multicut) Given an edge weighted graph $G = (V, E)$ and a set of source-sink pairs $\{(s_1, t_1), \cdots, (s_T, t_T\}$, the goal is to give a partitioning $\mathcal{P} = \{P_1, P_2, \cdots, P_{|\mathcal{P}|}\}$ of G such that all the source sink pairs are separated, and $\max_{1 \leq i \leq |\mathcal{P}|} \delta(P_i)$ is minimized.*

In min-max multicut, we do not force each part of the partitioning to have a terminal and there could be some parts without any terminals in the final solution. However, in the min-max multiway cut problem introduced by Svitkina and Tardos [72], each part needs to have exactly one terminal. We prove the following theorem for min-max multicut:

Theorem 2. *Given an edge weighted graph $G = (V, E)$ on n vertices, and a set of source sink pairs $S_G = \{(s_1, t_1), \cdots, (s_T, t_T)\}$, there exists a polynomial time algorithm which outputs a partitioning $\mathcal{P} = \{P_1, \cdots, P_{|\mathcal{P}|}\}$ of*

G, such that all the source sink pairs are separated and $\max_{1\leq i\leq|\mathcal{P}|}\delta(P_i)\leq \mathcal{O}(\sqrt{\log n\cdot\max\{\log(T),\log(k)\}})\cdot OPT$*; where OPT is the value of the optimum solution of min-max multicut, and k is the number of clusters in the optimum solution.*

We get improved approximation ratios for min-max correlation clustering, min-max multicut on graphs excluding a fixed minor.

Theorem 3. *Given an edge weighted graph G excluding $K_{r,r}$ minors, there exist polynomial time $\mathcal{O}(r^2)$-approximation algorithms for min-max correlation clustering and min-max multicut.*

Finally, we get improved approximation ratio for min-max correlation clustering on complete graphs.

Theorem 4. *Given an unweighted complete graph on the set of vertices V ($|V|=n$) such that each edge is labeled positive or negative, there exists a polynomial time algorithm which outputs a clustering $\mathcal{C}=\{C_1,\cdots,C_{|\mathcal{C}|}\}$ of G such that the disagreement on each $C_i\in\mathcal{C}$ is at most $14\cdot OPT$; where OPT is the maximum disagreement on each cluster in an optimal solution of min-max correlation clustering.*

High-Level Ideas

Most algorithms for correlation clustering with the global minimizing disagreement objective use a linear programming relaxation [54, 68, 69]. The recent work of Charikar, Gupta and Schwartz also use a similar linear programming relaxation for the vertex-wise min-max objective [19]. Surprisingly, these relaxations do not work for the min-max correlation clustering problem considered in this chapter. Indeed, simply obtaining a linear programming relaxation for the cluster-wise min-max objective looks challenging!

Bansal et al. [73] considered a semidefinite programming (SDP) based approximation algorithm for min-max k balanced partitioning and min-max multiway cut with k terminals. In their approach, instead of finding the entire solution in one shot, they obtain a single part at a time. It is possible to encode the same problem with a linear program albeit with a worse approximation guarantee. They use SDP rounding to obtain a part with low cut capacity and repeat the process until the parts produce a covering of all the vertices. By properly adjusting the weight of each part, the covering can be obtained efficiently. Finally, they convert the covering to partitioning.

The problem of extracting a single cluster of min-max correlation clustering can be captured by a semidefinite programming formulation. Here it

is not over a cut capacity objective, instead we need to simultaneously consider the intra-cluster negative edges as well as inter-cluster positive edges. Indeed, even for the global minimization objective, we are not aware of any good rounding algorithm based on SDP relaxation of correlation clustering. Therefore, rounding the SDP formulation directly looks difficult. To overcome this, we instead consider a new problem of *min-max multicut*. Demaine et al. [69] have shown an approximation preserving reduction between multicut and correlation clustering (for the global objective function). By solving the min-max multicut problem and then using the aforementioned reduction, we solve the min-max correlation clustering problem.

First, the reduction of Demaine et al. [69] is for the global objective, and an equivalence in global objective does not necessarily correspond to equivalency in local min-max objective. Fortunately, we could show indeed such an equivalency can be proven (Section 4.4). Thus, the "multicut" route seems promising as it optimizes over a cut objective. We consider obtaining each component of the min-max multicut problem, repeat this process to obtain a covering [73], and finally convert the covering to a partitioning.

The major technical challenge comes in rounding the SDP relaxation for the multicut instance where we seek to find a single component with good cut property. In order for the relaxation to be valid, we have to add new

constraints so that no source-sink pair (s_i, t_i) appears together. We also need to ensure that the component obtained satisfies a weight lower bound by assigning weights to each vertex. This is important in the next step when we wish to get a covering of all the vertices: we will decrease the weight of the vertices in the component recovered and again recompute the SDP relaxation with the same weight lower bound. This ensures the same component is not repeatedly recovered and a final covering can be obtained. To solve min-max multiway cut, Bansal et al. [73] need to separate k terminals. To do so, they can just guess which of the k terminals if any should appear in the current component with only $k+1$ guesses. For us, the number of such guesses would be 3^T where T is the number of source sink pairs since for every pair (s_i, t_i), either s_i or t_i or none would be part of the returned component. Since T could be $O(n^2)$ such a guessing is prohibitive. We need to come up with a new approach to address this issue.

We use an SDP relaxation to compute a metric on the graph vertices and add additional constraints to separate source sink pairs along with the spreading constraints from Bansal et al. [73] to recover a component of desired size. Next, we use the SDP separator technique introduced by Bansal et al. [73] to design a rounding algorithm that returns a set $S = \{S_1, S_2, \cdots, S_j\}$, such that for each $S_i \in S$, there are no source-sink pairs in S_i. Bansal et al. [73]

need to glue the sets in S and report it as a single component, since they wish to get a solution with specified number of components at the end. However, in min-max multicut problem, the number of components does not matter. Therefore, we do not need to union the sets in S, and as a result no source-sink violations happen.

It is possible to use a linear programming formulation for the detour via multicut and use LP-separators of Bansal et al. [73] in place of orthogonal separators and follow our algorithm. This would achieve a similar bound for min-max multicut and min-max correlation clustering in general graphs, but a much better bound of $\mathcal{O}(r^2 \cdot OPT)$ for graphs that exclude $K_{r,r}$ minors. Similarly, we use LP formulation of correlation clustering problem to devise a new algorithm for complete graphs.

4.3 Min-Max Multicut

Given a subset $S \subseteq V$, let $\delta(S)$ denote the number of edges with exactly one endpoint in S and let the number of source sink pairs (s_i, t_i) such that both s_i and t_i belong to S be $vio(S)$.

In order to prove Theorem 2, we first wish to find a set $S = \{S_1, \cdots, S_j\}$, such that $\forall S_i \in S, S_i \subseteq V$, and $\delta(S_i) \leq \mathcal{O}(\sqrt{\log n \cdot \max\{\log(|T|), \log(k)\}}) \cdot$

OPT, where OPT is the maximum number of cut edges on each part of the optimum partitioning for the min-max multicut problem on graph G, k is the number of clusters in the optimum solution which is guessed, T is the number of source-sink pairs, and n is the number of vertices in G. In addition, $\forall S_i \in S, vio(S_i) = \mathcal{O}(1)$.

Graph $G = (V, E)$ can have arbitrary edge weights, $w : E \to \mathbb{R}^+$. We assume graph $G = (V, E)$ is also a vertex-weighted graph, and there is a measure η on V such that $\eta(V) = 1$. This measure is used to get a covering of all the vertices. In Section 4.3, Theorem 5 is repeatedly applied to generate a family of sets that cover all the vertices. When a vertex is covered its weight is decreased so the uncovered vertices have a higher weight. Constraint $\eta(S) \in \eta(S) = \sum_{i=1}^{j} \eta(S_i) \in \left[H/4, 12H\right]$ makes sure the newly computed family of sets S has adequate coverage. Parameter $H \in (0, 1)$ is equal to $1/k$ where k is the number of parts in the optimum partitioning which we guess.

After getting a covering of all the vertices, in Section 4.3, it is explained how to convert a covering into a partitioning with the properties desired in Theorem 2. In order to prove Theorem 1, in Section 4.4, we show how a $\mathcal{O}(\sqrt{\log n \cdot \max\{\log(|T|), \log(k)\}})$-approximation algorithm for min-max multicut implies a $\mathcal{O}(\sqrt{\log n \cdot \max\{\log(|E^-|), \log(k)\}})$-approximation algorithm for min-max correlation clustering.

First, we prove the following theorem:

Theorem 5. *We are given an edge-weighted graph $G = (V, w)$, a set of source sink pairs S_G, a measure η on V such that $\eta(V) = 1$, and a parameter $H \in (0, 1)$. Assume there exists a set $S^* \subseteq V$ such that $\eta(S^*) \in [H, 2H]$, and $vio(S^*) = 0$. We design an efficient randomized algorithm to find a set S, where $S = \{S_1, \cdots, S_j\}$ satisfying $\forall S_i \in S, S_i \subseteq V$, $\eta(S) = \sum_{i=1}^{j} \eta(S_i) \in \big[H/4, 12H\big]$, and $\forall S_i \in S$, $vio(S_i) = 0$, and:*

$$\delta(S_i) \leq \mathcal{O}(\sqrt{\log n \cdot \max\{\log(T), \log(k)\}}) \cdot OPT$$

where $OPT = \arg\min \big\{\delta(S^*) : \eta(S^*) \in [H, 2H], \forall (s_i, t_i) \in S_G, |\{s_i, t_i\} \cap S^*| \leq 1\}$ and $|S_G| = T$.

In order to prove this theorem, we use the notion of $m-$orthogonal separators, a distribution over subsets of vectors, introduced by Chlamtac et al. [74] which is explained in the following:

Definition 3. *Let X be an ℓ_2^2 space (i.e. a finite collection of vectors satisfying ℓ_2^2 triangle inequalities with the zero vector) and $m > 0$. A distribution over subsets S of X is an $m-$orthogonal separator of X with probability scale $\alpha > 0$, separation threshold $0 < \beta < 1$, and distortion $D > 0$, if the following conditions hold:*

- $\forall u \in X, \Pr(u \in S) = \alpha \|u\|^2$
- $\forall u, v \in X$ *if* $\|u - v\|^2 \geq \beta \min\{\|u\|^2, \|v\|^2\}$ *then* $\Pr(u \in S$ *and* $v \in S) \leq \frac{\min\{\Pr(u \in S), \Pr(v \in S)\}}{m}$
- $\forall u, v \in X$, $\Pr(I_S(u) \neq I_S(v)) \leq \alpha D \cdot \|u - v\|^2$, *where* I_S *is the indicator function for the set* S.

Operator $\|.\|$ shows the ℓ^2 norm. Chlamtac et al. [74] proposed an algorithm for finding m-orthogonal separators.

Theorem 6. *[74] There exists a polynomial-time randomized algorithm that given an* ℓ_2^2 *space* X *containing* 0 *and a parameter* $m > 0$, *and* $0 < \beta < 1$, *generates an* $m-$*orthogonal separator with distortion* $D = \mathcal{O}_\beta(\sqrt{\log |X| \log m})$ *and* $\alpha \geq \frac{1}{poly(|X|)}$.

SDP Relaxation

In order to prove Theorem 5, we use the following SDP relaxation which is inspired by Bansal et al. [73] except for Constraints 4.5 and 4.6. In this relaxation, we assign a vector $\bar{v}$ for each vertex $v \in V$. The objective is to minimize the total weight of cut edges. The set of Constraints 4.2 are ℓ_2^2 triangle inequalities, and the set of Constraints 4.3 and 4.4 are ℓ_2^2 triangle inequalities with the zero vector. The set of Constraints 4.5 and 4.6 make sure

that for each source-sink pair (s_i, t_i), both s_i and t_i do not belong to S since both vectors $\bar{s_i}$ and $\bar{t_i}$ could not be $\mathbf{1}$ for some fixed unit vector simultaneously. Constraint 4.7 and the set of Constraints 4.8 make sure the returned subgraph has the desired size. Suppose now that we have approximately guessed the measure H of the optimal solution $H \leq \eta(S) \leq 2H$. We can ignore all vertices $v \in V$ with $\eta(v) > 2H$ since they do not participate in the optimal solution and thus write the set of Constraints 4.8. Constraints (4.9) are spreading constraints introduced by Bansal et al. [73] which ensure size of S is small.

$$\min \sum_{(u,v)\in E} w(u,v) \left\|\bar{u} - \bar{v}\right\|^2 \tag{4.1}$$

$$\left\|\bar{u} - \bar{w}\right\|^2 + \left\|\bar{w} - \bar{v}\right\|^2 \geq \left\|\bar{u} - \bar{v}\right\|^2 \qquad \forall u, v, w \in V \tag{4.2}$$

$$\left\|\bar{u} - \bar{w}\right\|^2 \geq \left\|\bar{u}\right\|^2 - \left\|\bar{w}\right\|^2 \qquad \forall u, w \in V \tag{4.3}$$

$$\left\|\bar{u}\right\|^2 + \left\|\bar{v}\right\|^2 \geq \left\|\bar{u} - \bar{v}\right\|^2 \qquad \forall u, v \in V \tag{4.4}$$

$$\left\|\bar{s_i} - \bar{t_i}\right\|^2 \geq \left\|\bar{s_i}\right\|^2 \qquad \forall (s_i, t_i) \in S_G \tag{4.5}$$

$$\left\|\bar{s_i} - \bar{t_i}\right\|^2 \geq \left\|\bar{t_i}\right\|^2 \qquad \forall (s_i, t_i) \in S_G \tag{4.6}$$

$$\sum_{v\in V} \left\|\bar{v}\right\|^2 \eta(v) \geq H \tag{4.7}$$

$$\left\|\bar{v}\right\|^2 = 0 \qquad \text{if } \eta(v) > 2H \tag{4.8}$$

$$\sum_{v\in V} \eta(v) \cdot \min\{\left\|\bar{u} - \bar{v}\right\|^2, \left\|\bar{u}\right\|^2\} \geq (1 - 2H) \left\|\bar{u}\right\|^2 \qquad \forall u \in V \tag{4.9}$$

Lemma 1. *Given* $S^* = \arg\min \left\{\delta(T) : \eta(T) \in [H, 2H], \forall (s_i, t_i) \in S_G, |\{s_i, t_i\} \cap T| \leq 1\right\}$, *the optimal value of SDP is at most* $\delta(S^*)$.

Proof. Consider the following solution for the SDP. For each vertex v, if $v \in S^*$ let $\bar{v} = \mathbf{1}$ for some fixed unit vector, and $\bar{v} = \mathbf{0}$ otherwise. We show this

gives a feasible solution for the SDP. Clearly triangle inequalities hold for this solution. Now we show that Constraints 4.5 hold for S^*. Consider a source-sink pair (s_i, t_i). Two cases might happen, either $\bar{s_i} = \mathbf{0}$ or $\bar{t_i} = \mathbf{0}$. If $\bar{s_i} = \mathbf{1}$ then $\bar{t_i}$ should be zero since s_i and t_i could not both belong to S^* and the constraint holds. If $\bar{s_i} = \mathbf{0}$ then RHS is 0 and the constraint holds. A similar argument shows the set of Constraints 4.6 hold as well. H was guessed such that $H \leq \eta(S^*) \leq 2H$ therefore Constraints 4.7 and Constraints 4.8 hold. Now we want to show the set of Constraints (4.9) hold. Consider a fixed vertex u. If $\bar{u} = \mathbf{0}$ then both sides of the spreading constraint become 0. If $\bar{u} = \mathbf{1}$ the spreading constraint equals $\eta(V \setminus S^*) \geq (1 - 2H)$ which holds since $H \leq \eta(S^*) \leq 2H$. Therefore S^* is a feasible solution and the objective value of SDP is at most $\delta(S^*)$. □

Approximation Algorithm

In this section, we prove Theorem 5. We propose an approximation algorithm which is inspired by Bansal et al.'s [73] algorithm for small-set expansion (SSE). However, there is a significant difference between our algorithm and theirs. In the SSE problem, one does not need to worry about separating source sink pairs.

First, we solve the SDP relaxation, and then proceed iteratively. In each iteration, we sample an $(32T \cdot k)$ - orthogonal separator S with $\beta = 1/2$ and return it (we repeatedly sample S, until a particular function[1] $f(S)$ has some positive value. Details are deferred to Section 4.3). Then, S is removed from graph G and the SDP solution, by zeroing the weight of edges incident on S (i.e. discarding these edges), and zeroing the SDP variables corresponding to vertices in S. The algorithm maintains the subsets of vertices removed so far in a set $U \subseteq V$, by initializing $U = \emptyset$, and then at each iteration by updating $U = U \cup \{S\}$. We keep iterating until $\eta(U) = \sum_{S_i \in U} \eta(S_i) \geq H/4$. After the last iteration, if $\eta(U) > H$, we output $F = S$ where S is computed in the last iteration. Otherwise, we put $F = U$. Note that in this case, $U = \{S_1, \cdots, S_{|U|}\}$.

Analysis

First, let's see what the effect of algorithm's changes to the SDP solution is. By zeroing vectors in S and discarding the edges incident on S, the SDP value may only decrease. Triangle inequalities, and the source-sink constraints still hold. Constraint $\sum_{v \in V} \|\bar{v}\|^2 \eta(v) \geq H$ will be violated due to zeroing some variables. However, since before the last iteration $\eta(U) \leq \frac{H}{4}$, the following

[1] defined later

constraint still holds:

$$\sum_{v \in V} \|\bar{v}\|^2 \eta(v) \geq \frac{3H}{4} \tag{4.10}$$

Next, we show the set of spreading constraints (4.9) will remain satisfied after removing S. Consider the spreading constraint for a fixed vertex u, two cases might happen:

Case 1: If $\exists S \in U$ such that $u \in S$, then u will be removed and $\|\bar{u}\| = 0$, the spreading constraint will be satisfied since RHS is 0.

Case 2: If $\nexists S \in U$ such that $u \in S$, the RHS will not change and we can show that $\min\{\|\bar{u} - \bar{v}\|^2, \|\bar{u}\|^2\}$ does not decrease. If $\nexists S' \in U$ such that $v \in S'$, then the term $\min\{\|\bar{u} - \bar{v}\|^2, \|\bar{u}\|^2\}$ does not change. If $\exists S' \in U$ such that $v \in S'$, then $\min\{\|\bar{u} - \bar{v}\|^2, \|\bar{u}\|^2\} = \|\bar{u}\|^2$ since $\|\bar{v}\| = 0$, and its value does not decrease.

Therefore, in both these cases, the spreading constraints will not be violated.

Lemma 2. *Let S be a sampled $(T \cdot k)$-orthogonal separator. Fix a vertex u. We claim that* $\Pr[\eta(S) \leq 12H \mid u \in S] \geq \frac{7}{8}$.

Proof. Consider a vertex u and let $A_u = \{v : \|\bar{u} - \bar{v}\|^2 \geq \beta \|\bar{u}\|^2\}$ and $B_u = \{v : \|\bar{u} - \bar{v}\|^2 < \beta \|\bar{u}\|^2\}$. Assume for now that $u \in S$. We show with high

probability $\eta(A_u \cap S)$ is small, and $\eta(B_u)$ is also small. Vertex u satisfies the spreading constraint. It is easy to see that:

$$(1-2H)\left\|u\right\|^2 \leq \sum_{v\in V} \eta(v)\cdot\min\{\left\|\bar{u}-\bar{v}\right\|^2, \left\|\bar{u}\right\|^2\} \leq \beta\left\|\bar{u}\right\|^2\eta(B_u)+\left\|\bar{u}\right\|^2\eta(A_u)$$

Since $\eta(V)=1$ and $A_u \cup B_u = V$, $\eta(A_u)+\eta(B_u)=1$, and $\beta = 1/2$ therefore:

$$(1-2H) \leq \beta\eta(B_u) + (1-\eta(B_u)) \tag{4.11}$$

$$\therefore \eta(B_u) \leq \frac{2H}{1-\beta} = 4H \tag{4.12}$$

Consider an arbitrary vertex $v \in A_u$ where $\|\bar{v}\| \neq 0$. By definition of A_u, $\left\|\bar{u}-\bar{v}\right\|^2 \geq \beta\left\|\bar{u}\right\|^2 \geq \beta\min\{\left\|\bar{u}\right\|^2, \left\|\bar{v}\right\|^2\}$. Therefore, by the second property of orthogonal separators and since we assumed $u \in S$, then $\Pr[v \in S \mid u \in S] \leq \frac{1}{32Tk} \leq \frac{1}{32k} \leq H$.

Now we show a bound for $\mathbb{E}[\eta(A_u \cap S) \mid u \in S]$:

$$\mathbb{E}[\eta(A_u \cap S) \mid u \in S] = \sum_{v\in A_u} \eta(v)\Pr[v \in S \mid u \in S] \leq H$$

Now, we want to bound $\Pr[\eta(S) \geq 12H \mid u \in S]$. The event $\{\eta(S) \geq 12H \mid u \in S\}$ implies the event $\{\eta(A_u \cap S) \geq 8H \mid u \in S\}$ since $\eta(B_u \cap S) \leq \eta(B_u) \leq 4H$.

(The second inequality holds by (4.12)). Now we are ready to complete the proof.

$$\Pr[\eta(S) \geq 12H | u \in S] \leq \Pr[\eta(A_u \cap S) \geq 8H | u \in S] \leq \frac{\mathbb{E}[\eta(A_u \cap S) \mid u \in S]}{8H} \leq \frac{H}{8H} = 1/8$$

We showed $\Pr[\eta(S) \geq 12H \mid u \in S] \leq 1/8$, therefore $\Pr[\eta(S) \leq 12H \mid u \in S] \geq 7/8$ and the proof is complete. □

Next, we upper bound $\delta(S)$. By the third property of orthogonal separators:

$$\mathbb{E}[\delta(S)] \leq \alpha D \cdot \sum_{(u,v)\in E} \|\bar{u} - \bar{v}\|^2 \cdot w(u,v) \leq \alpha D \cdot SDP$$

Where $D = \mathcal{O}_\beta(\sqrt{\log n \cdot \log(32T \cdot k)}) = \mathcal{O}(\sqrt{\log n \cdot \max\{\log(T), \log(k)\}})$. Note that $\beta = 1/2$. Consider the function f:

$$f(S) = \begin{cases} \eta(S) - \delta(S) \cdot \frac{H}{32D \cdot SDP} - 16vio(S) \cdot Hk & \text{if } S \neq \emptyset \text{ and } \eta(S) < 12H \\ 0 & \text{otherwise} \end{cases}$$

We wish to lower bound $\mathbb{E}[f(S)]$. First, we lower bound $\mathbb{E}[\eta(S)]$. As a result

of Lemma 2 and the first property of orthogonal separators:

$$\mathbb{E}[\eta(S)] = \sum_{u\in V} \Pr[u \in S \wedge \eta(S) < 12H] \cdot \eta(u)$$
$$= \sum_{u\in V} \Pr[\eta(S) < 12H \mid u \in S] \cdot \Pr[u \in S] \cdot \eta(u) \geq \sum_{u\in V} \frac{7\alpha\|\bar{u}\|^2\eta(u)}{8}$$

In the following we put a bound on $\mathbb{E}[vio(S)]$:

$$\mathbb{E}[vio(S)] = \sum_{1\leq i\leq T} \mathbf{1}(s_i \in S \wedge t_i \in S) \leq \sum_{1\leq i\leq T} \frac{\alpha \min\{\|\bar{s_i}\|^2, \|\bar{t_i}\|^2\}}{32Tk} \leq \frac{\alpha T}{32Tk} = \frac{\alpha}{32k}$$

Since $\mathbb{E}[vio(S)] \leq \frac{\alpha}{32k}$, $\mathbb{E}[\delta(S)] \leq \alpha D \cdot SDP$ and using Constraint 4.10:

$$\mathbb{E}[f(S)] \geq \sum_{u\in V} \frac{7\alpha\|\bar{u}\|^2\eta(u)}{8} - \alpha\cdot D\cdot SDP\cdot\frac{H}{32D\cdot SDP} - \frac{\alpha}{32k}\cdot 16Hk \geq \frac{7\alpha\frac{3H}{4}}{8} - \frac{\alpha H}{32} - \frac{\alpha H}{2} = \frac{1}{8}\alpha H$$

We have $f(S) \leq 2nH$ since $\|\bar{u}\| = 0$ whenever $\eta(u) > 2H$. Therefore, $\Pr[f(S) > 0] \geq \frac{\frac{1}{8}\alpha H}{2nH} = \Omega(\frac{\alpha}{n})$. So after $\mathcal{O}(n^2/\alpha)$ samples, with probability exponentially close to 1, the algorithm finds a set S with $f(S) > 0$. If $f(S) > 0$ then $\eta(S) \geq \delta(S) \cdot \frac{H}{32D\cdot SDP}$, therefore $\delta(S) \leq \frac{32D\cdot SDP\cdot\eta(S)}{H}$.

Additionally, $f(S) > 0$ implies $vio(S) \leq \frac{\eta(S)}{16Hk} < \frac{12}{16k} = \frac{3}{4k}$. The second inequality holds since $\eta(S) < 12H$. Since $k \geq 1$, $vio(S) < 1$ and hence none of the (s_i, t_i) pairs belong to the same cluster S.

Consider the two possible cases for the output F:

Case 1: $F = U = \{S_1, S_2, \cdots, S_{|U|}\}$, and $\eta(F) = \sum_{i=1}^{|U|} \eta(S_i)$. In this case, $\frac{H}{4} \leq \eta(F) \leq H$. The set U is a set of orthogonal separators and each $S_i \in U$ forms a separate part.

Case 2: $F = S$. In this case, let's show the last iteration of step 1 as $U = U_{old} \cup \{S\}$. We know $\eta(U) > H$, and $\eta(U_{old}) < \frac{H}{4}$, therefore $\eta(S) > 3H/4$. Also $f(S) > 0$ implies $\eta(S) \leq 12H$. Therefore, $3H/4 < \eta(S) \leq 12H$.

In both cases, $\frac{H}{4} \leq \eta(F) \leq 12H$.

We showed when a set $S_i \in U$ is sampled, $\delta(S_i) \leq \frac{32D \cdot SDP \cdot \eta(S_i)}{H}$. However, in the LHS of this inequality, edges like (u, v) where $u \in S_j, v \in S_i$ and $j < i$ are not considered. We can show $\sum_{j=1}^{i-1} \delta(S_j, S_i) \leq \sum_{j=1}^{i-1} \frac{32D \cdot SDP \cdot \eta(S_j)}{H} \leq 32D \cdot SDP$ since $\sum_{j=1}^{i-1} \eta(S_j) \leq H$. Therefore, $\delta(S_i) \leq \frac{32D \cdot SDP \cdot \eta(S_i)}{H} + \sum_{j=1}^{i-1} \delta(S_j, S_i) \leq \mathcal{O}(D \cdot SDP)$ since $\eta(S_i) \leq 12H$.

This completes the proof of Theorem 5.

The following corollary is implied from Theorem 5 and is used in the next section.

Corollary 2. *Given an edge-weighted graph $G = (V, w)$, a set of source sink pairs S_G, a measure η on V such that $\eta(V) = 1$, and a parameter τ, a set $S = \{S_1, \cdots, S_j\}$ could be found satisfying $\forall S_i \in S, S_i \subseteq V, vio(S_i) = 0$, and $\delta(S_i) \leq \mathcal{O}(\sqrt{\log n \cdot \max\{\log(T), \log(k)\}}) \cdot OPT$, where $OPT = \arg\min\{\delta(S^*) : \frac{\eta(S^*)}{\eta(V)} \geq \tau, vio(S^*) = 0\}$. In addition, $\eta(S) = \sum_{i=1}^{j} \eta(S_i) \geq \Omega(\tau \cdot \eta(V))$.*

Proof. The algorithm guesses $H \geq \tau$ such that $H \leq \eta(OPT) \leq 2H$. Guessing is feasible since $0 \leq \eta(OPT) \leq n \cdot \eta(u)$, where u is the weight of the heaviest element in OPT, and H can be chosen from the set $\{2^t\eta(u) : u \in V, t = 0, \cdots, \log(n)\}$ of size $\mathcal{O}(n\log(n))$. Theorem 5 is invoked with parameter H. The obtained solution S satisfies the properties of this corollary. To be more specific, by invoking Theorem 5, $\eta(S) = \sum_{i=1}^{j} \eta(S_i) \geq \frac{\tau}{4} \cdot \eta(V)$. □

Covering & Aggregation

Once we find F, we follow the multiplicative update algorithm of [73] with some minor modifications, to get a covering of all the vertices. Then, we use the aggregation step to convert the covering to a partitioning. This step is simpler than [73] since we are not required to maintain any size bound on the subgraphs returned after aggregation.

Theorem 7. *Given graph $G = (V, E)$ and T pairs of source and sink, running Algorithm 4 on this instance outputs a multiset $\mathcal{S}$ that satisfies the following conditions:*

- *for all $S \in \mathcal{S}$: $\delta(S) \leq D \cdot OPT$ where $D = \mathcal{O}(\sqrt{\log n \cdot \max\{\log(T), \log(k)\}}), vio(S) = 0$.*
- *for all $v \in V$, $\frac{|\{S \in \mathcal{S} : v \in S\}|}{|\mathcal{S}|} \geq \frac{1}{17kn}$, where k is the number of parts in the*

optimal solution which we guess.

Algorithm 4: Covering Procedure for Min-Max Multicut

Set $t = 1, S = \emptyset$ and $y^1(v) = 1\ \forall v \in V$;

Guess k, which is the number of parts in the optimal solution;

while $\sum_{v \in V} y^t(v) > \frac{1}{n}$ **do**

Find set $S^t = \{S_1, \cdots, S_j\}$ using Corollary 2, where $\tau = \frac{1}{k}$ and

$\forall v \in V, \eta(v) = y^t(v) / \sum_{v \in V} y^t(v)$;

$\mathcal{S} = S^t \cup \mathcal{S}$;

// Update the weights of the covered vertices;

for $v \in V$ **do**

Set $y^{t+1}(v) = \frac{1}{2} \cdot y^t(v)$ if $\exists S_i \in S^t$ such that $v \in S_i$, and

$y^{t+1}(v) = y^t(v)$ otherwise.;

Set $t = t + 1$;

return $\mathcal{S}$;

Proof. For an iteration t, let $Y^t = \sum_{v \in V} y^t(v)$. Consider the optimal solution $\{S_i^*\}_{i=1}^k$ to the min-max multicut problem. There exists at least a $S_j^* \in \{S_i^*\}_{i=1}^k$ with weight greater than or equal to the average ($y_t(S_j^*) \geq \frac{Y^t}{k}$), $vio(S_j^*) = 0$, and $\delta(S_j^*) \leq OPT$. Therefore by Corollary 2 where $H = \frac{1}{k}$, a set $S_t = \{S_1, S_2, \cdots, S_j\}$ could be found where $\forall S_i \in S_t$, $\delta(S_i) \leq \mathcal{O}(\sqrt{\log n \cdot \max\{\log(T), \log(k)\}}) \cdot OPT$, $vio(S_i) = 0$.

Now we show the second property of the theorem holds. Let ℓ denote

the number of iterations in the while loop. Let $|\{S \in \mathcal{S} : v \in S\}| = N_v$. By the updating rules $y^{\ell+1}(v) = 1/2^{N_v}$. Therefore $\frac{1}{2^{N_v}} = y^{\ell+1}(v) \leq 1/n$, which implies $N_v \geq \log(n)$. By Corollary 2, $y^t(S^t) \geq \frac{1}{4k} Y^t$. Therefore:

$$Y^{t+1} = Y^t - \frac{1}{2} y^t(S^t) \leq (1 - \frac{1}{8k}) Y^t$$

Which implies $Y^\ell \leq (1 - \frac{1}{8k})^{\ell-1} Y^1 = (1 - \frac{1}{8k})^{\ell-1} n$. Also $Y^\ell > 1/n$ therefore, $\ell \leq 1 + 16k \ln(n) \leq 17k \log(n)$. In each iteration t, the number of sets in S_t is at most n (since all the sets in S_t are disjoint), therefore $|\mathcal{S}| \leq 17kn \log(n)$, and the second property is proved. □

Now the covering of G is converted into a partitioning of G without violating min-max objective by much.

Theorem 8. *Given a weighted graph $G = (V, E)$, a set of source-sink pairs $(s_1, t_1), \cdots, (s_T, t_T)$, and a cover $\mathcal{S}$ consisting of subsets of V such that:*

- *$\forall v \in V$, v is covered by at least a fraction $\frac{c}{nk}$ of sets $S \in \mathcal{S}$, where k is the number of partitions of the optimum solution which we guessed in the covering section, and $c = 1/17$.*
- *$\forall S \in \mathcal{S}$, $\delta(S) \leq B$, $vio(S) = 0$.*

We propose a randomized polynomial time algorithm which outputs a partition $\mathcal{P}$ of V such that $\forall P_i \in \mathcal{P}$, $\delta(P_i) \leq 2B$, and $vio(P_i) = 0$.

Algorithm 5: Aggregation Procedure for Min-Max Multicut

Step 1: Sort sets in $\mathcal{S}$ in a random order: $S_1, S_2, \cdots, S_{|\mathcal{S}|}$. Let $P_i = S_i \setminus (\cup_{j<i} S_j)$.

Step 2: **while** *There is a set* P_i *such that* $\delta(P_i) > 2B$ **do**

  Set $P_i = S_i$ and for all $j \neq i$, set $P_j = P_j \setminus S_i$;

Proof. A similar proof to the one given by Bansal et al. [73] shows after step 2, for each $P_i \in \mathcal{P}$, $\delta(P_i) \leq 2B$. We start by analyzing Step 1. Observe that after Step 1, the collection of sets $\{P_i\}$ is a partition of V and $P_i \subseteq S_i$ for every i. Particularly, $vio(P_i) \leq vio(S_i)$. Note, however, that the bound $\delta(P_i) \leq B$ may be violated for some i since P_i might be a strict subset of S_i.

We finish the analysis of Step 1 by proving that $\mathbb{E}[\sum_i \delta(P_i)] \leq 2knB/c$. Fix an $i \leq |\mathcal{S}|$ and estimate the expected weight of edges $E(P_i, \cup_{j>i} P_j)$, given that the i^{th} set in the random ordering is S. If an edge (u, v) belongs to $E(P_i, \cup_{j>i} P_j)$, then $(u, v) \in E(S_i, V \setminus S_i) = E(S, V \setminus S)$ and both $u, v \notin \cup_{j<i} S_j$. For any edge $(u, v) \in \delta(S)$ (with $u \in S, v \notin S$), $\Pr((u, v) \in E(P_i, \cup_{j>i} P_j) \mid S_i = S) \leq \Pr(v \notin \cup_{j<i} S_j \mid S_i = S) \leq (1 - \frac{c}{nk})^{i-1}$, since v is covered by at least $\frac{c}{nk}$ fraction of sets in $\mathcal{S}$ and is not covered by $S_i = S$. Hence,

$$\mathbb{E}[w(E(P_i, \cup_{j>i} P_j)) \mid S_i = S] \leq (1 - \frac{c}{nk})^{i-1} \delta(S) \leq (1 - \frac{c}{nk})^{i-1} B$$

and $\mathbb{E}[w(E(P_i, \cup_{j>i} P_j)) \leq (1 - \frac{c}{nk})^{i-1} B$. Therefore:

$$\mathbb{E}\left[\sum_i \delta(P_i)\right] = 2 \cdot \mathbb{E}\left[\sum_i w(E(P_i, \cup_{j>i} P_j))\right] \leq 2 \sum_{i=0}^{\infty} (1 - \tfrac{c}{nk})^i B = 2knB/c$$

Now we want to analyze step 2. Consider potential function $\sum_i \delta(P_i)$, we showed after step 1, $\mathbb{E}\left[\sum_i \delta(P_i)\right] \leq 2knB/c$. We prove that this potential function reduces quickly over the iterations of Step 2, thus, Step 2 terminates after a small number of steps. After each iteration of Step 2, the following invariant holds: the collection of sets $\{P_i\}$ is a partition of V and $P_i \subseteq S_i$ for all i. Particularly, $vio(P_i) \leq vio(S_i)$. Using an uncrossing argument, we show at every iteration of the while loop in step 2, $\sum_i \delta(P_i)$ decreases by at least $2B$.

$$\begin{aligned}
&\delta(S_i) + \textstyle\sum_{j \neq i} \delta(P_j \setminus S_i) \leq \delta(S_i) + \sum_{j \neq i} \Big(\delta(P_j) + w(E(P_j \setminus S_i, S_i)) - w(E(S_i \setminus P_j, P_j))\Big) \\
&\leq \delta(S_i) + \textstyle\sum_{j \neq i} \Big(\delta(P_j)\Big) + w(E(V \setminus S_i, S_i)) - w(E(P_i, V \setminus P_i)) \\
&= \textstyle\sum_j \Big(\delta(P_j)\Big) + 2\delta(S_i) - 2\delta(P_i) \leq \sum_j \Big(\delta(P_j)\Big) - 2B
\end{aligned}$$

The above inequalities use the facts that $P_i \subseteq S_i$ for all i and that all the P_j's are disjoint. The second inequality uses the facts that $\sum_{j \neq i} w(E(P_j \setminus S_i, S_i)) = w(E(V \setminus S_i, S_i))$, and $\sum_{j \neq i} w(E(S_i \setminus P_j, P_j)) \geq w(E(P_i, V \setminus P_i))$, which hold since the collection of sets $\{P_i\}$ is a partition of V, and $P_i \subseteq S_i$.

In particular, $\sum_{j \neq i} w(E(S_i \setminus P_j, P_j)) \geq w(E(P_i, V \setminus P_i))$ holds since for each edge e if $e \in E(P_i, P_j)$ then $e \in E(S_i \setminus P_j, P_j)$. The last inequality holds since $\delta(S_i) \leq B$ and $\delta(P_i) > 2B$.

This proves that the number of iterations of the while loop is polynomially bounded and after step 2, $\delta(P_i) \leq 2B$ for each P_i.

In addition, since each P_i is a subset of S_i, $vio(P_i) \leq vio(S_i)$. Therefore $vio(P_i) = 0$. □

4.4 Analysis of Algorithm for Min-Max Correlation Clustering

In order to prove Theorem 1, we reduce a correlation clustering instance to a multicut instance. We follow the reduction shown by Demaine et al. [69]. They proved that the global objective multicut and correlation clustering are equivalent. However, equivalency of multicut and correlation clustering with respect to global objective does not immediately imply their equivalency with respect to min-max objective. In the following, first we mention the reduction, and then we show how a $\mathcal{O}(\sqrt{\log n \cdot \max\{\log(|T|), \log(k)\}})$-approximation algorithm for min-max multicut implies a $\mathcal{O}(\sqrt{\log n \cdot \max\{\log(|E^-|), \log(k)\}})$-approximation algorithm for min-max correlation clustering.

Given a graph $G = (V, E)$ which is an instance of correlation clustering, we construct a new graph $G' = (V', E')$ and a collection of source sink pairs $S_{G'} = \{\langle s_i, t_i \rangle\}$ as follows: Initially $V' = V$. For every negative edge $(u, v) \in E^-$ with weight $w(u, v)$, we add a new vertex uv to V' and a new edge (u, uv) to E' with weight $w(u, v)$. Also we add a source sink pair (v, uv) to $S_{G'}$. For every positive edge $(u, v) \in E^+$ with weight $w(u, v)$, we add (u, v) with weight $w(u, v)$ to E'. Now we have a multicut instance on G' with source sink pairs $S_{G'}$. Using Theorem 2, we find a partitioning $\mathcal{P} = \{P_1, P_2, \cdots, P_{|\mathcal{P}|}\}$ of G'. Next, we show how to convert $\mathcal{P}$ into a clustering $\mathcal{C}$ for graph G and prove Theorem 1.

In order to map a partitioning $\mathcal{P}$ into a clustering $\mathcal{C}$ for graph G, for each subset $P_i \in \mathcal{P}$, create a cluster C_i and for all $v \in V$, if $v \in P_i$, add v to C_i. We show the number of disagreements on each cluster $C_i \in \mathcal{C}$ ($cost(C_i)$) is at most the cut capacity of the corresponding subset $P_i \in \mathcal{P}$ ($\delta(P_i)$). Next, we prove this algorithm gives an $\mathcal{O}(\sqrt{\log n \cdot \max\{\log(|E^-|), \log(k)\}})$-approximation algorithm for min-max correlation clustering.

Lemma 10. *For all $P_i \in \mathcal{P}$ and the corresponding cluster C_i, $cost(C_i) \leq \delta(P_i)$.*

Proof. We show if C_i pays for an edge, then P_i will also pay for that edge. Consider an arbitrary edge (u, v), it could be either positive or negative.

Case 1: (u, v) is a positive edge. In this case if C_i is paying for (u, v), which

happens when one of u or v is in C_i and the other one is in $V \setminus C_i$, without loss of generality assume $u \in C_i$ and $v \in V \setminus C_i$, then $u \in P_i$ and $v \in V' \setminus P_i$. Therefore P_i will also pay for (u, v).

Case 2: (u, v) is a negative edge. In this case if C_i is paying for (u, v) then (u, v) is trapped inside C_i. Consider the corresponding multicut instance. In this instance, $u, v \in P_i$, there is a new vertex uv, a new edge (u, uv) and (v, uv) is a source-sink pair which implies $uv \in V' \setminus P_i$. Assume $uv \in P_j$, then the multicut solution pays for edge (u, uv) on both parts P_i and P_j. Therefore if C_i pays for a negative edge, the corresponding part in the multicut partitioning will also pay for that edge. □

Lemma 11. *$cost(C^*) \geq \delta(\mathcal{P}^*)$ where C^* is the optimum solution for the min-max correlation clustering on G and $\mathcal{P}^*$ is the optimum solution for the min-max multicut on G'.*

Proof. We construct a partitioning $\mathcal{P}$ of G' which separates all the source-sink pairs in G', in addition $cost(C^*) = cost(\mathcal{P})$. For each cluster C_i^*, construct a set P_i and $\forall v \in C_i^*$, add v to P_i. For all $uv \in V' \setminus V$ initially make them singleton clusters. It is easy to see that all source-sink pairs are separated in $\mathcal{P}$. Also for all positive edges in G, $\mathcal{P}$ and C^* are paying the same price. The only difference in the cost of $\mathcal{P}$ and C^* could happen for negative edges. Two cases might happen: First, consider a negative edge $(u, v) \in C_i^*$. In this case

C_i^* is paying for (u, v). In $\mathcal{P}$, $u \in P_i$ and uv is a singleton cluster. Edge (u, uv) is cut and P_i and the singleton cluster uv are paying for it. Therefore P_i and C_i^* are paying the same price $w(u, v)$ for edge (u, v). The singleton cluster uv is also paying the same price $w(u, v)$ for that edge. In addition the singleton cluster uv is not paying for any other edge which means cost of it is at most cost of C_i^*. The other case is when a negative edge (u, v) is between clusters, i.e. $u \in C_i^*, v \in C_j^*$. Therefore C^* is not paying for (u, v) but $\mathcal{P}$ is paying for that edge since (u, uv) is cut in $\mathcal{P}$. In this case we move uv into the part P_i. By doing that, source-sink pair (uv, v) is still separated since $uv \in P_i, v \in P_j$. Also since (u, uv) is not cut anymore, $\mathcal{P}$ and C^* are paying the same price for edge (u, v).

Therefore:

$$cost(C^*) = cost(\mathcal{P})$$

Where $cost(C^*)$ is the maximum number of disagreements on each cluster of C^* and $cost(\mathcal{P})$ is the maximum number of cut edges on each part of $\mathcal{P}$.

Also $cost(\mathcal{P}) \geq cost(\mathcal{P}^*)$. Therefore $cost(C^*) \geq \delta(\mathcal{P}^*)$ and the proof is complete. □

Now we are ready to prove Theorem 1.

Theorem 1. *Given an edge weighted graph $G = (V, E)$ on n vertices such*

that each edge is labeled positive or negative, there exists a polynomial time algorithm which outputs a clustering $\mathcal{C} = \{C_1, \cdots, C_{|\mathcal{C}|}\}$ *of* G *such that the disagreement on each* $C_i \in \mathcal{C}$ *is at most* $\mathcal{O}(\sqrt{\log n \cdot \max\{\log(|E^-|), \log(k)\}}) \cdot OPT$*; where* OPT *is the maximum disagreement on each cluster in an optimal solution of min-max correlation clustering,* k *is the number of clusters in the optimum solution, and* $|E^-|$ *denotes the number of negative edges in* G*.*

Proof. Let C^* be the optimum solution for the min-max correlation clustering on G, and $\mathcal{P}^*$ be the optimum solution for the min-max multicut on G'. By Theorem 2 we can find a partitioning $\mathcal{P}$ of G such that $cost(\mathcal{P}) \leq \mathcal{O}(\sqrt{\log n \cdot \max\{\log(|T|), \log(k)\}}) \cdot cost(\mathcal{P}^*)$. We convert partitioning $\mathcal{P}$ into a clustering C as it was explained earlier in this section. Therefore:

$$cost(\mathcal{P}^*) \leq cost(C^*) \leq cost(C) \leq cost(\mathcal{P})$$

The first inequality holds by Lemma 11. The third inequality holds by Lemma 10. Since $cost(\mathcal{P}) \leq \mathcal{O}(\sqrt{\log n \cdot \max\{\log(|T|), \log(k)\}}) \cdot cost(\mathcal{P}^*)$ and the number of source-sink pairs in the min-max multi-cut is equal to the number of negative edges in the min-max correlation clustering instance, it could be seen that:

$$cost(C) \leq \mathcal{O}(\sqrt{\log n \cdot \max\{\log(|E^-|), \log(k)\}}) \cdot cost(C^*)$$

□

4.5 Min-Max Correlation Clustering and Min-Max Multicut in Minor-Closed Graph Families

In this section, we show improved results for min-max correlation clustering and min-max multicut in minor-closed graph families. The procedure is almost similar to what we did for general graphs. We wish to solve min-max correlation clustering on a weighted graph $G = (V, E)$ excluding a fixed minor $K_{r,r}$. First, we do the same reduction proposed by Demaine et al. [69] that we mentioned in Section 4.4 to get a multicut instance G'. In the following, we show G' excludes $K_{r,r}$ minors as well. After that we prove Theorem 9 which is similar to Theorem 5.

Lemma 3. *If G is excluding a fixed minor $K_{r,r}$ then G' also excludes minor $K_{r,r}$.*

Proof. We get G' from G by deleting some edges and adding some new vertices and connecting them to exactly one vertex of G. It could be seen if G was excluding minor $K_{r,r}$ after these operations G' will be excluding minor $K_{r,r}$ as well. □

Theorem 9. *Given an edge-weighted graph $G = (V, w)$ excluding $K_{r,r}$ minors, a set of source sink pairs S_G, a measure η on V such that $\eta(V) = 1$ and a parameter $H \in (0, 1)$, there is an efficient algorithm to find a set S, where $S = \{S_1, \cdots, S_j\}$ satisfying $\forall S_i \in S, S_i \subseteq V$, $\eta(S) = \sum_{i=1}^{j} \eta(S_i) \in \big[H/4, 12H\big]$ and $\forall S_i \in S$, $vio(S_i) = \mathcal{O}(1)$:*

$$\delta(S_i) \leq \mathcal{O}(r^2) \cdot \min \big\{\delta(T) : \eta(T) \in [H, 2H], \forall (s_i, t_i) \in S_G, |\{s_i, t_i\} \cap T| \leq 1\big\}$$

In order to prove Theorem 9, we write an LP which is analogue with the SDP we used for general graphs. We use some ideas Bansal et al. [73] used to write an LP for min-max k-partitioning problem in minor-closed graph families. As Bansal et al. [73] explain, for every vertex $u \in V$ we introduce a variable $x(u)$ such that $0 \leq x(u) \leq 1$. For every pair of vertices $u, v \in V$ we introduce a variable $z(u, v) = z(v, u)$ taking values in $\big[0, 1\big]$. The intended integral solution corresponding to a set $S \subseteq V$ has $x(u) = 1$ if $u \in S$ and $x(u) = 0$ otherwise; $z(u, v) = |x(u) - x(v)|$. One could think of $x(u)$ as

an analogue of $\|\bar{u}\|^2$ and of $z(u,v)$ as an analogue of $\|\bar{u}-\bar{v}\|^2$ in the SDP relaxation. In order to prove Theorem 9, we use a notion of LP-separators

$\min \sum_{(u,v)\in E} w(u,v)z(u,v)$	
$z(u,v)+z(v,w) \geq z(u,w)$	$\forall u,v,w \in V$
$\lvert x(u)-x(v)\rvert \leq z(u,v)$	$\forall u,v \in V$
$x(u)+x(v) \geq z(u,v)$	$\forall u,v \in V$
$\lvert x(s_i)-x(t_i)\rvert \geq x(s_i)$	$\forall$ source-sink pair (s_i,t_i)
$\lvert x(s_i)-x(t_i)\rvert \geq x(t_i)$	$\forall$ source-sink pair (s_i,t_i)
$\sum_{v\in V} x(v)\eta(v) \geq H$	
$x(v)=0$	if $\eta(v) > 2H$
$\sum_{v\in V} \eta(v)\cdot \min\{x(u), z(u,v)\} \geq (1-2H)x(u)$	$\forall u \in V$
$x(u), z(u,v) \in [0,1]$	$\forall u,v \in V$

introduced by Bansal et al. [73].

Definition 4. *(LP separator) Given a graph $G=(V,E)$ and numbers $\{x(u), z(u,v)\}_{u,v\in V}$, a distribution over subsets $S \subseteq V$ is an LP separator with distortion $D \geq 1$, probability scale $\alpha > 0$ and separation threshold $\beta \in (0,1)$ if:*

- *for all $u \in V$, $\Pr(u \in S) = \alpha \cdot x(u)$*
- *for all $u,v \in V$ with $z(u,v) \geq \beta \cdot \min\{x(u), x(v)\}$, $\Pr(u \in S$ and $v \in S) = 0$*
- *for all $(u,v) \in E$ we have $\Pr(I_S(u) \neq I_S(v)) \leq \alpha D \cdot z(u,v)$, where I_S is*

the indicator function for the set S.

The following theorem was proved by Bansal et. al [73].

Theorem 10. *[73] Given a graph $G = (V, E)$ that excludes $K_{r,r}$ minors, numbers $\{x(u), z(u, v)\}_{u,v \in V}$ satisfying the first three constraints of LP and parameter $\beta \in (0, 1)$, there exists an algorithm which returns an LP separator with distortion $D = \mathcal{O}(r^2/\beta)$, probability scale $\alpha = \Omega(1/|V|)$ and separation threshold β.*

By replacing the SDP relaxation with the LP relaxation and the orthogonal separators with LP separators, Theorem 9 could be proved. The rest of procedure is same as what we did for general graphs. At the end, Theorem 3 can be proved.

Bansal et. al [73] showed for genus g graphs, an LP separator with distortion $\mathcal{O}(\log(g))$ can be obtained. By following a similar approach an $\mathcal{O}(\log(g))$-approximation for min-max multicut and min-max correlation clustering on genus g graphs can be obtained.

4.6 Min-Max Correlation Clustering on Complete Graphs

In order to prove Theorem 4, we assume the existence of a measure η on V such that $\eta(V) = 1$. This measure is used to generate a covering of all the vertices by leveraging Theorem 11 multiple times. When a vertex is covered, the corresponding weight is decreased so that the uncovered vertices get a higher weight (Using the multiplicative algorithm of [73]), followed by partitioning. The covering and partitioning algorithms are same as that of general graphs (Section 4.3). First, we prove the following theorem, followed by the covering and partitioning algorithm:

Theorem 11. *We are given an unweighted complete graph G on the set of vertices V ($|V| = n$) such that each edge is labeled positive or negative, a measure η on V such that $\eta(V) = 1$, and a parameter $H \in (0, 1)$. Assume there exists a set $T \subseteq V$ such that $\eta(T) \geq H$. We design an efficient algorithm to find a set $\mathcal{S}$, where $\mathcal{S} = \{S_1, \cdots, S_j\}$ satisfying $\forall S_i \in \mathcal{S}, S_i \subseteq V$, $\eta(\cup S_i) \geq H$, and:*

$$cost(S_i) \leq 7 \cdot \min \left\{ cost(T) : \eta(T) \geq H \right\}$$

To prove Theorem 11, we use the following integer linear program (ILP)

that tries to solve for T with minimum $cost(T)$ such that $\eta(T) \geq H$.

$$\min \sum_{(u,v)\in E^+} d(u,v) + \sum_{(u,v)\in E^-} (\max\{x(u), x(v)\} - d(u,v)) \tag{4.13}$$

$$d(u,w) + d(w,v) \geq d(u,v), \qquad \forall u,v,w \in V \tag{4.14}$$

$$|x(u) - x(v)| \leq d(u,v), \qquad \forall u,v \in V \tag{4.15}$$

$$d(u,v) \leq x(u) + x(v), \qquad \forall u,v \in V \tag{4.16}$$

$$x(u) + x(v) + d(u,v) \leq 2, \ \forall u,v \in V \tag{4.17}$$

$$\sum_{v\in V} x(v)\eta(v) \geq H \tag{4.18}$$

$$x(u), d(u,v) \in \{0,1\}, \qquad \forall u,v \in V \tag{4.19}$$

In this LP formulation, every node u has a variable $x(u)$ and every edge has a disagreement $d(u,v)$ $\forall u,v \in V$. The constraints 4.14 to 4.16 are the triangle inequality constraints and 4.17 ensures that at most two of the three variables can have the value of 1. The last constraint ensures that $\eta(T) \geq H$.

Lemma 4. *Given* $T^* = \arg\min \{cost(T) : \eta(T) \geq H\}$*, the optimal value of Integer LP is at most* $cost(T^*)$*.*

Proof. Consider a candidate solution, such that $x(u) = 1$ if $u \in T^*$ and 0 otherwise. Hence, $d(u,v) = 1$ only when $x(u) = 1$ and $x(v) = 0$ or vice versa. This variable assignment, satisfies the triangle inequalities and also

$\sum_{v \in V} \eta(v)x(v) = \eta(T^*) \geq H$. The contribution of the edges to the objective function is as follows:

1. $u, v \in T^*$ implies $x(u) = x(v) = 1$ and $d(u, v) = 0$. The contribution of (u,v) is 0 if $(u, v) \in E^+$ and 1 otherwise.
2. $u, v \notin T^*$ implies $x(u) = x(v) = 0$ and $d(u, v) = 0$. The contribution of any edge (u, v) when $x(u) = x(v) = 0$ is 0.
3. $u \in T^*,\ v \notin T^*$ implies $x(u) = 1$ and $x(v) = 0$, hence $d(u, v) = 1$. The contribution of (u, v) is 0 if $(u, v) \in E^-$ and 1 otherwise.

This shows that the objective function captures the number of positive edges within T^* and negative edges to nodes outside T^*, which is equal to $cost(T^*)$. Hence, the optimal solution of this integer program has objective value at most $cost(T^*)$. □

Approximation Algorithm

We consider LP relaxation of the integer program with constraints 4.19 modified to $x(u), d(u, v) \in [0, 1], \forall u, v \in V$ and use Algorithm 6 to solve for T. We solve the LP relaxation by guessing a node in the optimal cluster. For every guess $u \in V$, we add a constraint $x(u) = 1$ in the above LP relaxation and identify the corresponding optimal fractional solution. Suppose d_u and x_u

is the corresponding optimal fractional solution with objective value o_u when u is the chosen guess. We sort these objective values in non-decreasing order to get a sorted list $\mathcal{O} = \{o_1, \ldots, o_{|V|}\}$ such that o_i is the optimal objective value of the LP relaxation when $u_i \in V$ is chosen as a guess. We process the sorted list to identify the smallest index λ such that $\sum_{j<\lambda} \eta(u_j) < H \leq \sum_{j\leq\lambda} \eta(u_j)$ and consider the set of these guesses, $\Gamma = \{u_i, i \leq \lambda\}$.

Firstly, the objective value $o_j,\ \forall j \leq \lambda$ is less than the optimal value of the integral objective function ($o_i \leq OPT,\ i \leq \lambda$; See Lemma 5). Secondly, for each guess $u_i \in \Gamma$, we run the rounding Algorithm 7 to construct an integer solution ($S_i \supseteq \{u_i\}$) which generates a 7-approximation of the corresponding fractional solution ($cost(S_i) \leq 7o_i$; See Lemma 6). This guarantees that each of the integer solution returned by Algorithm 6 is a 7-approximation of the optimal solution to the integer LP. Additionally, $\Gamma \subseteq \cup S_i$ and $\sum_{i\leq\lambda} \eta(u_i) \geq H$

ensures that the $\eta(\cup S_i) > H$. This completes the proof of Theorem 11.

Algorithm 6: Generate Covering

for $u_i \in V$ **do**

Let $o_{u_i}, d_{u_i}, x_{u_i}$ be the solution on solving the LP relaxation with an additional constraint $x(u_i) = 1$

Sort $\{o_u : u \in V\}$ in non-decreasing order to generate a sorted list:

$\{o_1, \ldots, o_{|V|}\}$, where o_i corresponds to the guess u_i

Let $\lambda \leftarrow \min_t : \sum_{i=1}^{t} \eta(u_i) \geq H$ and $\Gamma \leftarrow \{u_i : i \leq \lambda\}$

$\mathcal{S} \leftarrow \phi$

for $u_i \in \Gamma$ **do**

$S_i \leftarrow$ Use Algorithm 7 to round the LP solution, (d_{u_i}, x_{u_i})

$\mathcal{S} \leftarrow \mathcal{S} \cup \{S_i\}$

return $\mathcal{S}$

Now, we prove the following lemma's :

Lemma 5. *For every guess u_j, $j \leq \lambda$, the optimal solution of the LP relaxation $o_j \leq OPT$, where OPT is the optimal integral solution of the integer program considered.*

Proof. Let C denote the optimal integral solution of the Integer LP i.e. $x(v) = 1$, $\forall v \in C$ and 0 otherwise. Consider the LP relaxation when u_i, $i \leq \lambda$ is guessed. If $u_i \in C$, then C is a valid solution to the LP relaxation. Hence the objective value of the LP relaxation, $o_i \leq OPT$.

Suppose $\exists i \leq \lambda$ such that $u_i \notin C$. In this case, $\sum_{j<i} \eta(u_j) < H$ because $i \leq \lambda$. Hence there must exist k such that $k > i$ and $u_k \in C$, because

$\sum_{u \in C} \eta(u) \geq H$. Since, the objective values o_i's are arranged in non-decreasing order of objective value, $o_i \leq o_k$ and since $u_k \in C$, $o_k \leq OPT$. Hence $o_i \leq OPT$. □

For every guess $u \in \Gamma$, we show that the cluster returned by the rounding Algorithm 7 is 7-approximation of the optimal objective value of the corresponding LP relaxation. Hence, we will get a candidate solution for each guess. Below, we show the rounding algorithm and the corresponding approximation ratio.

Rounding Algorithm for a particular guess

Our rounding algorithm is motivated by the ball growing approach in [68]. We consider a ball of radius 2/7 (say T) around the guessed vertex and try to construct a cluster based on the total fractional disagreements of the vertices in T. If the total fractional disagreements are larger than 1/7 fraction of the number of vertices in the ball, it outputs a singleton cluster with the guessed vertex. On the other hand, if the total disagreements are lower than

1/7 fraction, it outputs the complete ball T along with the guess.

Algorithm 7: Rounding Algorithm for a guess u

$T = \{w \in V - \{u\} : d(u, w) \leq \frac{2}{7}\}$

if $\sum_{w \in T} d(u, w) \geq |T|/7$ **then**
  Output the cluster $\{u\}$

else
  Output the cluster $\{u\} \cup T$

Lemma 6. *Algorithm 7 identifies a cluster C such that the integral disagreements of C is 7-approximation of the corresponding fractional disagreements.*

Proof. We consider two different cases based on the output of the algorithm. For each case, we show that the integral contribution of an edge (or a combination of edges) is less than 7 times the fractional contribution of the corresponding edge (or corresponding combination of edges).

Notice that, constraint 4.15 implies that $x(u) - x(v) = 1 - x(v) \leq d(u, v)$ and, constraint 4.17 implies $x(v) \leq 2 - x(u) - d(u, v) = 1 - d(u, v)$ hence $x(v) = 1 - d(u, v) \geq 1 - 2/7 = 5/7$

Case 1: Only the node u is output as the cluster. In this case, the integral contribution to the objective is the set of positive neighbors of u. Consider the edge (u, v) such that $d(u, v) > 2/7$. In this case, the integral contribution is less than 7/2 times the fractional disagreement of that edge. When $d(u, v) \leq 2/7$, the integral contribution of those edges is at most $|T|$. Also, since $\sum_{w \in T} d(u, w) \geq |T|/7$, this means that the integral contribution is less

than 7 times the fraction of fractional contribution.

Case 2: When a cluster $\{u\} \cup T$ is returned. In this case, there are two sets of mistakes.

- **The negative edges within the cluster.** In this case, the contribution of negative edge (v, w), $v, w \in T$ to fractional disagreements is $\max\{x(v), x(w)\} - d(v, w) \geq x(v) - d(v, u) - d(u, w) \geq x(v) - 2 \cdot \frac{2}{7} \geq 5/7 - 4/7 = 1/7$.
- **The positive edges to nodes outside the cluster.** For the positive edges, let's consider a node outside the cluster, $z \notin T \cup \{u\}$. If $d(u, z) \geq 3/7$, then $d(v, z) \geq d(u, z) - d(u, v) \geq 3/7 - 2/7 = 1/7$

  If $2/7 < d(u, z) \leq 3/7$, we do the following: The total contribution of z towards the integral component of the cluster objective is $|P|$ where P is the set of positive edges between the nodes of $T \cup \{u\}$ with z and the number of negative edges incident on z is $|Q| = |T| + 1 - |P|$. The

fractional contribution of the edges incident on z is

$$\sum_{w \in P} d(w,z) + \sum_{w \in Q} (\max\{x(w), x(z)\} - d(w,z))$$

$$\geq \sum_{w \in P} (d(u,z) - d(u,w)) + \sum_{w \in Q} (x(w) - d(u,w) - d(u,z))$$

$$\geq d(u,z)|P| + \sum_{w \in Q} (x(w) - d(u,z)) - \sum_{w \in P \cup Q} (d(u,w))$$

$$\geq d(u,z)|P| + |Q|(5/7 - d(u,z)) - \sum_{w \in \{u\} \cup T} (d(u,w))$$

$$\geq d(u,z)|P| + |Q|(5/7 - d(u,z)) - \frac{|P| + |Q|}{7}$$

This equation is a linear function in $d(u,z)$. So, we evaluate its values at the end points of the line to identify min and max.

- When $d(u,z) = 2/7$, it evaluates to $|P|/7 + \frac{2}{7}|Q| > |P|/7$
- When $d(u,z) = 3/7$, it evaluates to $\left(\frac{2}{7}\right)|P| + \frac{1}{7}|Q| > \frac{1}{7}|P|$

This shows that the total integral disagreements of positive edges with any node $z \notin T$ is less than 7 times the fractional disagreements of corresponding edges. Hence, the approximation ratio of Algorithm 7 is 7.

Covering and Partitioning

We use the same covering algorithm that uses the multiplicative weights algorithm from [73] along with the partitioning strategy to generate non-overlapping clusters. For completeness, we present the modified theorem statements for the complete graphs case.

Theorem 12. *Given a complete graph, running Algorithm 4*[2] *on the instance outputs a multiset $\mathcal{S}$ that satisfies the following conditions:*

- $\forall S \in \mathcal{S}$

$$cost(S) \leq 7 \cdot OPT$$

- $\forall v \in V$,

$$\frac{|\{S \in \mathcal{S} : v \in S\}|}{|\mathcal{S}|} \geq \frac{1}{5nk}$$

Proof. Same as Proof of Theorem 7 □

The covering generated by Algorithm 4 is converted into a partitioning using Algorithm 5. The following result from Section 4.3 is used to bound to

[2]For complete graphs, the multiset S^t in line 4 is generated using Algorithm 6

approximation ratio of the generated partitions.

Theorem 13. *Given a complete graph and a cover $\mathcal{S}$ consisting of subsets of V such that:*

- *$\forall v \in V$, v is covered by at least $\frac{c}{nk}$ sets $S \in \mathcal{S}$ where k is the number of partitions in the optimum solution which we guessed in the covering section and $c \in (0, 1]$ and $cost(S) \leq B$*

We propose a randomized algorithm which outputs a partition $\mathcal{P}$ of V such that $\forall P_i \in \mathcal{P}, cost(P_i) \leq 2B$.

Proof. Same as Theorem 8 for the positive edges. We can ignore the negative edges in this analysis as the cost of negative edges can never increase on splitting a cluster. □

Using Theorem 12 and 13, we can generate a 14 approximation of the local correlation clustering problem for complete graphs.

□

4.7 Authors

This Chapter was written by Saba Ahmadi, Sainyam Galhotra, Samir Khuller, Barna Saha, and Roy Schwartz. A preliminary version of this work

was written by Saba Ahmadi, Samir Khuller, and Barna Saha and appeared at the Integer Programming and Combinatorial Optimization (IPCO) 2019 [66].

5 The Strategic Perceptron

5.1 Introduction

In machine learning, *strategic classification* deals with the problem of learning a classifier when the learner relies on data that is provided by strategic agents [75, 76]. For example, consider deciding eligibility of individuals for employment or education. In order to be considered eligible, individuals may engage in activities that do not truly change their qualifications, but affect the decision made. In the aforementioned settings, these activities include job or college applicants carefully crafting their application materials and investing in interview or test preparations. In these scenarios, by using information about the classifier, individuals alter their features artificially by a limited amount to achieve their desirable outcome.

Strategic classification is particularly challenging in the online setting, where data points arrive in an arbitrary sequence, because the way that points

manipulate may depend (in a discontinuous way) on the current classifier, and there is no useful source of unmanipulated data. More specifically, consider a standard online learning setting as follows. Individuals arrive one at a time, and based on the individual's features, the classifier predicts the individual as positive or negative. The learner is then told the correct classification and may update its classifier for the next round. The learner's goal is to minimize the number of mistakes made. Performing the same procedure in the strategic setting brings in several challenges. First, since the learner does not observe the *true* features, the update is done based on the individual's *manipulated* features. Therefore, at each point in time, the current classifier is built from manipulated data the learner has observed in the past. Second, each individual reacts to the *current* classifier. This means that the individuals' behaviors change over time and may be different from behavior of previous individuals with similar features. Moreover, because data arrives in an arbitrary order, there is no way to collect a "representative sample" of unmanipulated data by, say, classifying all examples as negative for an initial period. Finally, manipulation behavior may be a discontinuous function of the classifier's parameters: if an individual's cost to manipulate is slightly less than the benefit of being classified as positive then it will do so, but if it is slightly greater then it will not. Due of these issues, as we will show, standard learning algorithms that

would make a limited number of mistakes in non-strategic settings may end up cycling and making unbounded number of mistakes; even if there exists a perfect classifier they may not find one.

Another challenge in online strategic classification is when the learner is unaware of the manipulation costs, which determine the extent to which agents will manipulate their features to achieve a positive classification. In this case, on top of estimating the individuals' real attributes based on the observed data, the learner also needs to estimate the costs. Unreasonable estimate of costs may lead to poor performance by the learner as the learner may not be able to distinguish if a classification mistake is due to an improper classifier or improper estimate of costs. This failure to distinguish correctly may lead to deterioration of the classifier and divergence from the optimal solution.

We study an online linear classification problem when the individuals are strategic. To isolate the effect of manipulation, we focus on finding a linear classifier when the unmanipulated data is linearly separable; i.e., the feature space is divided into two half spaces: with *positive* data points in one and *negative* data points in the other, and a nonzero margin between them. When individuals can manipulate, in each step, the arriving individual wishes to be classified positively. If the individual's feature vector $\mathbf{z}$ is not classified as positive with the true attributes, they may choose to suffer a cost and pretend

to have a feature vector $\mathbf{x}$. More specifically, we consider utility-maximizing individuals, where utility is defined as value minus cost, who receive value 1 for being classified as positive and 0 for being classified as negative. We then consider two classes of cost functions: ℓ_2 costs (where cost is proportional to the Euclidean distance moved) and weighted ℓ_1 costs (where the cost of reaching a destination is the sum of separate costs paid in each coordinate direction). The ℓ_2 case represents settings where individuals when manipulating can take actions that affect multiple attributes. The ℓ_1 case represents settings where there is a specific action associated with each attribute. Note that in both cases, even though the *unmanipulated* data is linearly separable, the observed *manipulated* data points may no longer be separable.

Our Techniques and Results The main contribution of this paper is solving the problem of online learning of linear separators in the strategic setting, making a bounded number of mistakes when the unmanipulated data is linearly separable by a nonzero margin. To do this, we build on and adapt the classic Perceptron algorithm [77], redesigning it to work in various strategic settings. This classic algorithm makes a bounded number of mistakes in the nonstrategic case when positive and negative data points are linearly separable. However, as mentioned earlier, in the strategic case it may cycle indefinitely

(much like gradient descent for finding a Nash equilibrium) and make an unbounded number of mistakes; see Examples 1 and 2. Our main technique is to carefully design *surrogate* data points and feed them as the observed data to the algorithm. The role of the surrogate is to ensure that the algorithm is able to make positive progress each time it makes a mistake; however, defining it requires extra care. In particular, while it is not hard to show we can compute the *direction* that data points may have manipulated in, we can never be sure exactly how far (and we are particularly interested in the case that the amount by which data points can manipulate is large compared to the margin of separation). Another adaptation is to use a positive threshold for the dot product with the classifiers weight vector for a point to be classified positive.

Making use of the Perceptron algorithm, surrogate data points, and a positive linear threshold is central in all the algorithms designed in this paper. However, additional ideas are needed to handle subtleties of each specific setting. For example, for weighted ℓ_1 costs we need to take extra steps to make the manipulation direction unique and in line with the true classifiers weight vector, and in the unknown costs setting, we need to distinguish if the cost estimates are above or below the true costs. Another case is when the separating hyperplane does not cross the origin. In this case, the classic approach is to just add a fake coordinate in which each example has value 1, and then apply

the Perceptron algorithm to those extended data points. However, when data is given by strategic agents, this reduction breaks down and we need to apply different ideas. There are also some results that hold for the non-strategic case that we do not know how to achieve, such as obtaining a mistake bound proportional to the hinge-loss of the best separator when data is not perfectly separable; for this setting we show examples where our algorithm fails and propose it as an open problem.

The main contributions of this paper are:

- We give an online learning algorithm robust to manipulation that finds a linear classifier in a bounded number of mistakes with the knowledge of costs. The number of mistakes is not much larger than the standard Perceptron bound in the non-strategic case for ℓ_2 costs and is reasonably bounded in other settings as well, see Theorems 10, 11 and 13.
- We give an online learning algorithm that generalizes the previous algorithm to unknown costs with a bounded number of mistakes. See Theorem 12.
- We generalize the algorithm for known ℓ_2 costs to the case of heterogeneous agents whose utility functions differ by a limited amount and give an online learning algorithm with bounded number of mistakes. See Corollary 3.

Related Work The first studies on strategic classification focused on the offline setting; i.e., where the agents' true features come from a distribution. Brückner and Scheffer [75] and later Hardt et al. [76], formalized the strategic classification problem as a Stackelberg competition between a learner and an agent. They assume the learner has access to the distribution of agents true features and their cost functions; and use this information to design near-optimal classifiers.

Dong et al. [78] initiated the study of strategic classification in the online setting where the learner does not know the distribution of agents' true features or their cost functions. A key difference between [78] and this paper is the assumption on the objective of the agents: we consider agents that wish to be classified as positive, whereas [78] considers agents that wish to increase their dot-product with the hypothesis vector no matter how they are classified. Based on this assumption, in [78] the agents behaviors are continuous in the hypothesis vector. However, in our model, a small change in the hypothesis vector can cause a drastic (discontinuous) change in agents' behavior. More particularly, as a consequence of agents objective and utility structure, each agent can manipulate by a limited amount. If the classification hyperplane is closer than this amount, the agent would manipulate to be classified as positive; however, if it is slightly farther, the agent stays stationary. This dis-

continuity in the agents behavior is common in mechanism design and occurs in other problems such as pricing and auction design.

Chen et al. [79] also study an online learning problem where agents can manipulate by a bounded distance. However, while there are similarities between the setting studied in [79] and our ℓ_2 cost model, explained in more detail in Section 5.2, [79] does not consider a fixed utility model and instead considers a regret term that is worst-case over agents that can manipulate by some bounded distance. As a result, their regret term may be arbitrarily high when the observed positions of positive and negative data points are inseparable, even if the unmanipulated points are linearly separable. Our algorithms, in contrast, can handle this inseparability during the learning procedure and make a bounded number of mistakes.

The goal of the papers mentioned so far is accuracy or minimizing loss. There are also papers that consider other objectives. Hu et al. [80] focus on a fairness objective and raise the issue that different populations of agents may have different manipulation costs. Braverman and Garg [81], by introducing noise in their classification, design algorithms where agents with different costs are better off not manipulating which tackles the fairness issue. Milli et al. [82] state that the accuracy that strategic classification seeks leads to a raised bar for agents who naturally are qualified and puts a burden on them to

prove themselves. Kleinberg and Raghavan [83], Haghtalab et al. [84], Alon et al. [85], Bechavod et al. [86], Shavit et al. [87], and Miller et al. [88] focus on models in which the policy maker is interested in choosing a rule which incentivizes agent(s) to invest their effort into features that truly improve their qualification.

Organization of the Paper. Section 5.2 introduces the model and provides examples where the original Perceptron algorithm makes an unbounded number of mistakes. Sections 5.3 and 5.4 study the case where the cost of manipulation is known: Section 5.3 focuses on ℓ_2 costs and Section 5.4 on weighted ℓ_1 costs. Section 5.5 studies the unknown costs model. Section 5.6 studies the generalization of known manipulation costs to heterogeneous agents that have slightly different costs. Section 5.7 extends the results of Sections 5.3 and 5.4 where the separator of the unmanipulated data points crosses the origin to the general case. Finally, conclusions and some open problems are presented in Section 5.8.

5.2 Model and Preliminaries

In Section 5.2, we formally define our model. In Section 5.2, we overview the non-strategic setting. In Section 5.2, we provide examples where the orig-

inal Perceptron algorithm makes an unbounded number of mistakes.

Model

We study an online classification problem in which a series of examples in $\mathbb{R}^d$ arrive one at a time. We think of examples as corresponding to d observable features of individuals who wish to be classified as positive. They have the ability to manipulate their observable features at some cost. Let $\mathbf{z}_t$ denote the t^{th} example before manipulation, and $\mathbf{x}_t$ denote the observed t^{th} example. We assume there exists a vector $\mathbf{w}^*$, such that for each unmanipulated positive example $\mathbf{z}_t$ we have $\mathbf{z}_t^T\mathbf{w}^* \geq 1$, and for each unmanipulated negative example $\mathbf{z}_t$ we have $\mathbf{z}_t^T\mathbf{w}^* \leq -1$; i.e., a linear separator of margin $\gamma = 1/|\mathbf{w}^*|$. We use $|\mathbf{w}|$ to indicate the ℓ_2 norm of $\mathbf{w}$.

We assume individuals are utility maximizers, where utility is defined as value minus cost. Individuals have value 1 for being classified as positive, and 0 for being classified as negative. More formally, an agent with true coordinates $\mathbf{z}_t$ will move to $\mathbf{x}_t = \arg\max_x[value(\mathbf{x}) - cost(\mathbf{z}_t, \mathbf{x})]$ where $value(\mathbf{x}) = 1$ if $\mathbf{x}$ is classified positive by the current classifier and $value(\mathbf{x}) = 0$ if $\mathbf{x}$ is classified negative, and $cost(\mathbf{z}_t, \mathbf{x})$ refers to the cost of manipulation from $\mathbf{z}_t$ to $\mathbf{x}$. This implies if the agent can manipulate their features at cost at most

1 to change their classification from negative to positive, then they will do so in the cheapest way possible, otherwise they will not.

We consider two settings for cost of manipulation. In the first setting, $cost(\mathbf{z}_t, \mathbf{x})$ is proportional to the ℓ_2 distance of the two points $\mathbf{z}_t$ and $\mathbf{x}$; i.e., $cost(\mathbf{z}_t, \mathbf{x}) = c\sqrt{\sum_{i=1}^{d}(\mathbf{x}_i - \mathbf{z}_{t,i})^2}$, where c is the cost per unit of movement. We define $\alpha = 1/c$ as the maximum amount data points would be willing to move to achieve a positive classification.[1] We assume $0 \leq \alpha \leq R$ where $R = \max_t |\mathbf{z}_t|$. In the second setting, $cost(\mathbf{z}_t, \mathbf{x})$ is a weighted ℓ_1 metric, such that $cost(\mathbf{z}_t, \mathbf{x}) = \sum_{i=1}^{d} c_i|\mathbf{x}_i - \mathbf{z}_{t,i}|$. Similarly we define $\alpha_i = 1/c_i$ as the maximum amount data points would move along the i^{th} coordinate vector $\mathbf{e}_i$, where $0 \leq \alpha_i \leq R$. We consider both scenarios of known and unknown costs. In the unknown ℓ_2 costs we don't assume knowledge of c, and in unknown weighted ℓ_1 costs we do not assume knowledge of $c_1, \ldots, c_d$.

Generalizations

For the majority of the paper we study the above model. In Sections 5.6 and 5.7, we then present and analyze several generalizations. Section 5.6 studies heterogeneous agents with ℓ_2 costs where the costs per unit of movement

[1] For convenience we assume that if an agent is indifferent, i.e., its distance to the decision boundary is exactly α, then it will manipulate. Note that Chen et al. [79] also consider a model where individuals can move in a ball of fixed radius from their real position. However, they do not focus on a specific utility model.

(defined previously as c) for agents are slightly different. More particularly, we study the case where the maximum amount the agent arriving at time t can move, α_t (i.e. $1/c_t$, where c_t is the cost per unit of movement for agent t), is in the interval $[\alpha_{\min}, \alpha_{\max}]$ where $0 \leq \alpha_{\max} - \alpha_{\min} \leq \gamma/2$. The algorithm does not have access to α_t but knows the interval. Section 5.7 studies the case where the separating hyperplane of the unmanipulated data does not cross the origin. More particularly, there exists a separator $\mathbf{z}^T\mathbf{w}^* + b = 0$, such that for a positive example $\mathbf{z}_t$, $\mathbf{z}_t^T\mathbf{w}^* + b \geq 1$ and for a negative example $\mathbf{z}_t$, $\mathbf{w}^{*T}\mathbf{z}_t + b \leq -1$.

Non-Strategic Setting and the Perceptron Algorithm

As a reminder for the reader we provide the classical Perceptron algorithm here. This algorithm classifies all points with $\mathbf{x}_t^T\mathbf{w} \geq 0$ as positive, and the rest as negative; updating $\mathbf{w}$ when it makes a mistake. The total number of mistakes made by the algorithm is upper bounded by $R^2|\mathbf{w}^*|^2$.

Extension 1 (Perceptron with separator not crossing the origin)**.** *A classic extension of Algorithm 8 to the case where examples are linearly separable, but not by a separator passing through the origin, is to create an extra "fake"*

Algorithm 8: Perceptron Algorithm

$\mathbf{w} \leftarrow \mathbf{0}$;
for $t = 1, 2, \cdots$ **do**
  Given example $\mathbf{x_t}$, predict $sgn(\mathbf{x}_t^T \mathbf{w})$;
  if *the prediction was a mistake* **then**
    if $\mathbf{x_t}$ *was* $+$ **then** $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}_t$;
    if $\mathbf{x}_t$ *was* $-$ **then** $\mathbf{w} \leftarrow \mathbf{w} - \mathbf{x}_t$;

coordinate. Specifically, assume there exists a separator $\mathbf{x}^T \mathbf{w}^* + b = 0$*, such that for a positive example* $\mathbf{x}_t$*,* $\mathbf{x}_t^T \mathbf{w}^* + b \geq 1$ *and for a negative example* $\mathbf{x}_t$*,* $\mathbf{w}^{*T} \mathbf{x}_t + b \leq -1$*. Then Algorithm 8 is extended by adding an extra coordinate of value* 1 *to each example* $\mathbf{x}_t$*, replacing* $\mathbf{x}_t$ *with* $(\mathbf{x}_t, 1)$*. The bias term* b *is absorbed into* $\mathbf{w}^*$ *by adding an additional coordinate to* $\mathbf{w}^*$*, i.e. replacing* $\mathbf{w}^*$ *with* $(\mathbf{w}^*, b)$*. Now, for the positive examples,* $\mathbf{x}_t^T \mathbf{w}^* \geq 1$*, and for the negative examples* $\mathbf{x}_t^T \mathbf{w}^* \leq -1$*, and Algorithm 8 can be used as before.*

Failure of the Perceptron Algorithm in Strategic Settings

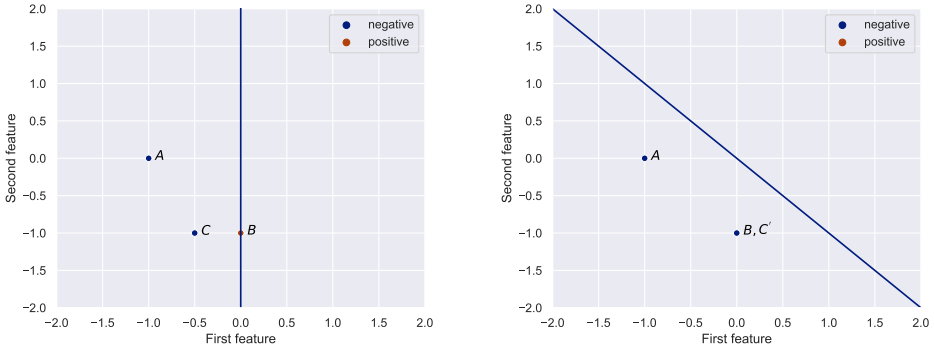
The Perceptron algorithm may make unbounded number of mistakes in the models considered in this paper even when a perfect classifier exists. The following example illustrates this in a setting with ℓ_2 cost.

Example 1. *Consider three examples* $A = (-1, 0)$*,* $B = (0, -1)$*, and* $C =$

$(-0.5, -1)$ *where* A *is negative,* B *is positive, and* C *is negative. Suppose that* $\alpha = 0.5$*. The following scenario of arrival of these examples makes the standard Perceptron algorithm (Algorithm 8) cycle between two classifiers and make an unbounded number of mistakes. Suppose* A *is the first example to arrive, then individuals* B *and* C *arrive respectively and repeatedly. After arrival of* A*,* $\mathbf{w} = (1, 0)$*.* B *does not need to manipulate as it is classified positive with the current classifier. However* C *manipulates to point* $(0, -1)$ *and the algorithm mistakenly classifies it as positive. As a consequence,* $\mathbf{w}$ *will be updated to* $(1, 0) - (0, -1) = (1, 1)$*. With the new classifier,* B *cannot manipulate to be classified positive because it has distance* $\sqrt{2}/2$ *from the decision boundary. So,* B *is misclassified as negative, causing an update to* $\mathbf{w} = (1, 1) + (0, -1) = (1, 0)$ *and the scenario repeats.*

Note that Example 1 shows that the standard Perceptron algorithm can fail even if there exists a classifier that is perfect in the presence of manipulation.In this example, the classifier given by $\mathbf{w} = (1, 0.5)$ works perfectly for the three points as B can manipulate to be classified positive but A and C cannot. The main reason the algorithm fails despite existence of a perfect classifier, is that the behavior of individuals *depends on the classifier* we are currently using and this can cause the algorithm to cycle indefinitely.

The failure of the Perceptron algorithm is not restricted to the ℓ_2 costs



(a) Classic Perceptron correctly classifies B after update of $\mathbf{w}$ to $(1, 0)$. However, C now manipulates to the same location as B, which will cause a mistake and an update.

(b) After C manipulates to $C' = (0, -1)$, it is misclassified as positive and $\mathbf{w}$ is updated to $(1, 1)$. B cannot manipulate with current classifier because it is at distance $\sqrt{2}/2$ from the boundary.

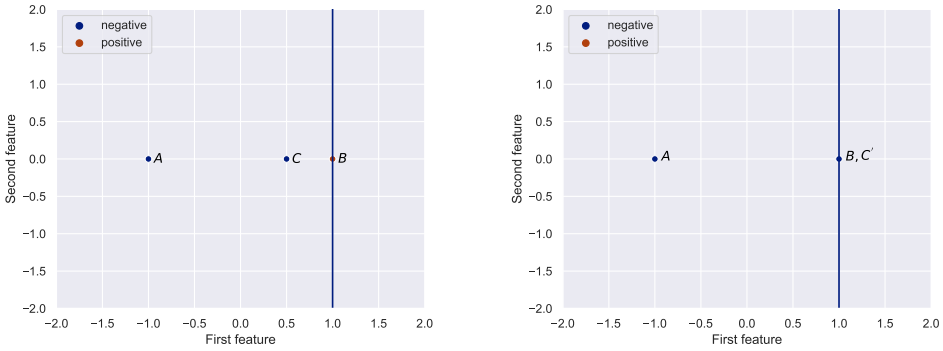
Figure 5.1: Example 1 shows the classic Perceptron algorithm can make an unbounded number of mistakes in the strategic setting.

model. Example 1 with $\alpha = (0.6, 0)$ makes an unbounded number of mistakes in the ℓ_1 costs model as well.

The Perceptron algorithm as described above uses a threshold of 0. One may wonder if the usual extension to non-zero thresholds (Extension 1) might solve the strategic learning problem. In particular, any linearly separable dataset is still linearly separable in the presence of manipulation, by simply shifting the target separator by α. However, the example below shows that this extension also fails when the data points are strategic.

Example 2. *Consider three examples* $A = (-1, 0)$, $B = (1, 0)$, *and* $C = (0.5, 0)$ *where* A *and* C *are negative and* B *is positive. Let* $\alpha = 0.5$. *Suppose*

A is the first example to arrive, then individuals B and C arrive respectively and repeatedly. After arrival of A, the separator is $1x_1 + 0x_2 - 1 \geq 0$. *B does not need to manipulate as it is classified positive with the current classifier as shown in Figure 5.2(a). However* $C = (0.5, 0)$ *manipulates to* $(1, 0)$ *and the algorithm mistakenly classifies it as positive as shown in Figure 5.2(b). As a consequence, the separator is updated to* $0x_1 + 0x_2 - 2 \geq 0$. *With the new classifier, B is misclassified as negative but does not manipulate. The separator is then updated to* $1x_1 + 0x_2 - 1 \geq 0$, *and the process repeats indefinitely. Therefore the classifier keeps cycling and never correctly classifies the data even when a linear separator exists.*



(a) The Perceptron algorithm updates **w** after misclassifying A. When B arrives it correctly classifies it.

(b) After C manipulates to $C' = (1, 0)$, it is now misclassified as positive by the current classifier and **w** and *bias* are updated accordingly.

Figure 5.2: Example 2 shows the non-zero threshold Perceptron algorithm can make an unbounded number of mistakes in the strategic setting.

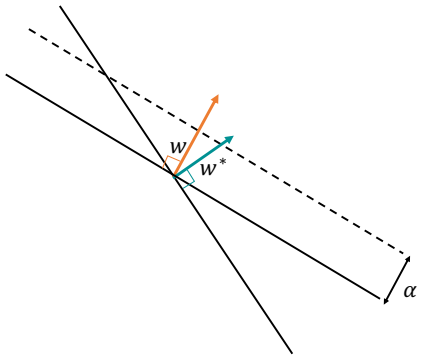


Figure 5.3: Strategic Perceptron with known manipulation cost. The dashed line represents the manipulation hyperplane discussed in Observation 1. The margin of width α is the forbidden region, discussed in Observation 2.

5.3 Known ℓ_2 Costs

In this section, we provide an algorithm for the ℓ_2 costs setting. At a high level, there are two main ideas to modify and generalize the Perceptron algorithm for this setting. The first modification is raising the bar for a point to be classified as positive. Previously, a nonnegative dot product with the current classifier (a threshold of 0), sufficed for positive classification. However, in the new algorithm, the threshold is a strictly positive value depending on the cost of manipulation. The second modification is using a *surrogate* for the data points when the classifier updates. Interestingly, we only need to use a surrogate for negative points, and in this case the surrogate is a projection of the point in the opposite direction of manipulation, detected by the algorithm.

Overview of Algorithm 9. This algorithm is a generalization of the Perceptron algorithm which we call *strategic Perceptron*. The algorithm starts by predicting all points as positive until it makes a mistake. Note that during this period, individuals do not have incentive to manipulate. From that point on, the algorithm classifies all points with $\mathbf{x}_t^T\mathbf{w}/|\mathbf{w}| - \alpha \geq 0$ as positive, and the rest as negative. Whenever the algorithm makes a mistake, the predictor $\mathbf{w}$ is updated with a surrogate value, $\tilde{\mathbf{x}}_t$, defined below.

Definition 5 ($\tilde{\mathbf{x}}_t$, surrogate data point in ℓ_2 setting)**.** *We define surrogate data point, $\tilde{\mathbf{x}}_t$, as follows.*

$$\tilde{\mathbf{x}}_t = \begin{cases} \mathbf{x}_t - \alpha\frac{\mathbf{w}}{|\mathbf{w}|}, & \text{if } \mathbf{x}_t \text{ is } - \text{ and } \frac{\mathbf{x}_t^T\mathbf{w}}{|\mathbf{w}|} = \alpha; \\ \mathbf{x}_t, & \text{if } \mathbf{x}_t \text{ is } + \text{ and } \frac{\mathbf{x}_t^T\mathbf{w}}{|\mathbf{w}|} = \alpha; \\ \mathbf{x}_t, & \text{if } \frac{\mathbf{x}_t^T\mathbf{w}}{|\mathbf{w}|} > \alpha \text{ or } \frac{\mathbf{x}_t^T\mathbf{w}}{|\mathbf{w}|} \leq 0. \end{cases}$$

Observation 1 (manipulation hyperplane)**.** *In Algorithm 9, $\mathbf{x}_t$ is a manipulated example only if $\mathbf{x}^T\mathbf{w}/|\mathbf{w}| = \alpha$. The reason is as follows. In order to maximize utility, individuals move data points in direction of $\mathbf{w}$ and move the point the minimum amount to be classified as positive. Therefore, if with true features they are classified as negative, they only need to move to the line with dot product equal to α and moving to any other location contradicts with utility*

Algorithm 9: Strategic Perceptron for ℓ_2 costs

$\mathbf{w} \leftarrow \mathbf{0}$;
for $t = 1, 2, \cdots$ **do**
  Given example $\mathbf{x}_t$:
  if $|\mathbf{w}|$ *is* 0 **then**
    predict $+$;
    if *the prediction was a mistake* **then** $\mathbf{w} \leftarrow \mathbf{w} - \mathbf{x}_t$;
  else
    predict $sgn(\frac{\mathbf{x}_t^T \mathbf{w}}{|\mathbf{w}|} - \alpha)$;
    if *the prediction was a mistake and* $\mathbf{x}_t$ *was* $+$ **then** $\mathbf{w} \leftarrow \mathbf{w} + \tilde{\mathbf{x}}_t$;
    if *the prediction was a mistake and* $\mathbf{x}_t$ *was* $-$ **then** $\mathbf{w} \leftarrow \mathbf{w} - \tilde{\mathbf{x}}_t$;

maximizing. In other words:

$$\mathbf{x}_t = \begin{cases} \mathbf{z}_t + \left(\alpha - \frac{\mathbf{z}_t^T \mathbf{w}}{|\mathbf{w}|}\right) \frac{\mathbf{w}}{|\mathbf{w}|}, & \text{if } 0 \leq \frac{\mathbf{z}_t^T \mathbf{w}}{|\mathbf{w}|} \leq \alpha; \\ \mathbf{z}_t, & \text{otherwise.} \end{cases}$$

Observation 2 (forbidden region)**.** *No observed data point* $\mathbf{x}_t$ *will satisfy* $0 < \mathbf{x}_t^T \mathbf{w}/|\mathbf{w}| < \alpha$*, and therefore* $\tilde{\mathbf{x}}_t$ *does not need to be defined for* $0 < \mathbf{x}_t^T \mathbf{w}/|\mathbf{w}| < \alpha$*. The reason is that any such data point must either have manipulated to that position or not. If it manipulated, the manipulation was not rational since it did not help the data point to get classified as positive. If it did not manipulate, this was not rational either since the data point has a distance less than* α *from the classifier.*

We show Algorithm 9 makes at most $(R + \alpha)^2 |\mathbf{w}^*|^2$ mistakes. First, we

need to prove the following lemmas hold.

Lemma 12. *For any positive data point* $\mathbf{x}_t$, $\tilde{\mathbf{x}}_t^T \mathbf{w}^* \geq 1$, *and for any negative data point* $\mathbf{x}_t$, $\tilde{\mathbf{x}}_t^T \mathbf{w}^* \leq -1$. *Also, throughout the execution of Algorithm 9,* $\mathbf{w}^T \mathbf{w}^* \geq 0$.

Proof. The proof uses induction. First, we show after the first update of the algorithm $\mathbf{w}^T \mathbf{w}^* > 0$. Second, we show if at the end of step $t-1$, $\mathbf{w}^T \mathbf{w}^* \geq 0$, then at step t, $\tilde{\mathbf{x}}_t^T \mathbf{w}^* \geq 1$ for positive points, and $\tilde{\mathbf{x}}_t^T \mathbf{w}^* \leq -1$ for negative points. Finally, we show if $\mathbf{w}^T \mathbf{w}^* \geq 0$ at the end of step $t-1$, and $\tilde{\mathbf{x}}_t^T \mathbf{w}^* \geq 0$ for positive points, and $\tilde{\mathbf{x}}_t^T \mathbf{w}^* \leq 0$ for negative points, then $\mathbf{w}^T \mathbf{w}^* \geq 0$ at the end of step t.

The first step is straight-forward. Initially, $\mathbf{w} = 0$. While $\mathbf{w} = 0$, we have $\mathbf{x}_t^T \mathbf{w} = 0$, and arriving examples get classified positively. The first mistake occurs when a negative example $\mathbf{x}_t$ arrives and gets classified as positive. In this case, $\mathbf{w}$ gets updated to $\mathbf{w} - \mathbf{x}_t$. Since $\mathbf{x}_t^T \mathbf{w}^* \leq -1$, we conclude $(\mathbf{w} - \mathbf{x}_t)\mathbf{w}^* > 0$.

The second step is more involved. By definition of the surrogate values, for any points such that $\mathbf{x}_t^T \mathbf{w}/|\mathbf{w}| \neq \alpha$, we have $\tilde{\mathbf{x}}_t = \mathbf{x}_t$. By Observation 1, these points are not manipulated, i.e., $\mathbf{x}_t = \mathbf{z}_t$. This implies $\tilde{\mathbf{x}}_t = \mathbf{z}_t$ and therefore the claim holds. Thus, we only need to argue for the points on the hyperplane $\mathbf{x}_t^T \mathbf{w}/|\mathbf{w}| = \alpha$. Consider such data points. For the positive data

points, we have, $\tilde{\mathbf{x}}_t = \mathbf{x}_t = \mathbf{z}_t + \beta \cdot \mathbf{w}/|\mathbf{w}|$, where $0 \leq \beta \leq \alpha$. Therefore, $\tilde{\mathbf{x}}_t^T \mathbf{w}^* = \mathbf{z}_t^T \mathbf{w}^* + \beta \cdot \mathbf{w}^T \mathbf{w}^*/|\mathbf{w}| \geq \mathbf{z}_t^T \mathbf{w}^* \geq 1$. The first inequality holds since by assumption of this step, $\mathbf{w}^T \mathbf{w}^* \geq 0$. On the other hand, for the negative data points we have $\tilde{\mathbf{x}}_t = \mathbf{x}_t - \alpha \cdot \mathbf{w}/|\mathbf{w}|$, where $\mathbf{x}_t = \mathbf{z}_t + \beta \cdot \mathbf{w}/|\mathbf{w}|$ and $0 \leq \beta \leq \alpha$. This implies $\tilde{\mathbf{x}}_t = \mathbf{z}_t + (\beta - \alpha) \cdot \mathbf{w}/|\mathbf{w}|$. By multiplying with $\mathbf{w}^*$, we get $\tilde{\mathbf{x}}_t^T \mathbf{w}^* = \mathbf{z}_t^T \mathbf{w}^* + (\beta - \alpha) \cdot \mathbf{w}^T \mathbf{w}^*/|\mathbf{w}| \leq \mathbf{z}_t^T \mathbf{w}^* \leq -1$.

The final step is again straight-forward. Whenever $\mathbf{w}$ is updated, for positive points, $\mathbf{w}$ gets updated to $\mathbf{w} + \tilde{\mathbf{x}}_t$, where both $\mathbf{w}$ and $\tilde{\mathbf{x}}_t$ have nonnegative dot product with $\mathbf{w}^*$. For negative points, $\mathbf{w}$ gets updated to $\mathbf{w} - \tilde{\mathbf{x}}_t$, where $\mathbf{w}$ has a nonnegative and $\tilde{\mathbf{x}}_t$ has a negative dot product with $\mathbf{w}^*$. □

Lemma 13. *When Algorithm 9 makes a mistake on a positive example* $\mathbf{x}_t$, $\tilde{\mathbf{x}}_t^T \mathbf{w} \leq 0$*; and when it makes a mistake on a negative example* $\mathbf{x}_t$, $\tilde{\mathbf{x}}_t^T \mathbf{w} \geq 0$.

Proof. The algorithm makes a mistake on a positive example only if $\mathbf{x}^T \mathbf{w}/|\mathbf{w}| < \alpha$. By Observation 2, for no points, $0 < \mathbf{x}^T \mathbf{w}/|\mathbf{w}| < \alpha$. Therefore, for any positive example that the algorithm makes a mistake on, $\mathbf{x}^T \mathbf{w} \leq 0$. By Definition 5, $\tilde{\mathbf{x}}_t = \mathbf{x}_t$ for all positive examples. Therefore, $\mathbf{x}^T \mathbf{w} \leq 0$ implies $\tilde{\mathbf{x}}^T \mathbf{w} \leq 0$. For negative examples, the algorithm makes a mistake only if $\mathbf{x}^T \mathbf{w}/|\mathbf{w}| \geq \alpha$. If the inequality is strict, i.e., $\mathbf{x}^T \mathbf{w}/|\mathbf{w}| > \alpha$, by Definition 5, $\tilde{\mathbf{x}}_t = \mathbf{x}_t$, and therefore $\tilde{\mathbf{x}}_t^T \mathbf{w} \geq 0$. If $\mathbf{x}^T \mathbf{w}/|\mathbf{w}| = \alpha$, again using Definition 5, we have $\tilde{\mathbf{x}}^T \mathbf{w} = 0$. □

Next, we show the following theorem holds which gives a bound on the number of mistakes. Proof of the following theorem is along the lines of the proof of the classic Perceptron algorithm.

Theorem 10. *Algorithm 9 makes at most* $(R+\alpha)^2|\mathbf{w}^*|^2$ *mistakes in the strategic setting with known* ℓ_2 *costs, when the unmanipulated data points* $\mathbf{z}_t$ *satisfy* $\mathbf{z}_t^T\mathbf{w}^* \geq 1$ *for positive examples and* $\mathbf{z}_t^T\mathbf{w}^* \leq -1$ *for negative examples, and* $R = \max_t |\mathbf{z}_t|$.

Proof. We keep track of two quantities, $\mathbf{w}^T\mathbf{w}^*$ and $|\mathbf{w}|^2$. First, we show that each time we make a mistake, $\mathbf{w}^T\mathbf{w}^*$ increases by at least 1. If we make a mistake on a positive example then,

$$(\mathbf{w} + \tilde{\mathbf{x}}_t)^T\mathbf{w}^* = \mathbf{w}^T\mathbf{w}^* + \tilde{\mathbf{x}}_t^T\mathbf{w}^* \geq \mathbf{w}^T\mathbf{w}^* + 1;$$

where the last inequality holds by Lemma 12. Similarly, if we make a mistake on a negative example,

$$(\mathbf{w} - \tilde{\mathbf{x}}_t)^T\mathbf{w}^* = \mathbf{w}^T\mathbf{w}^* - \tilde{\mathbf{x}}_t^T\mathbf{w}^* \geq \mathbf{w}^T\mathbf{w}^* + 1.$$

Next, on each mistake we claim that $|\mathbf{w}|^2$ increases by at most $(R+\alpha)^2$. If we

make a mistake on a positive example $\mathbf{x}_t$, then we have:

$$(\mathbf{w}+\tilde{\mathbf{x}}_t)^T(\mathbf{w}+\tilde{\mathbf{x}}_t) = |\mathbf{w}|^2 + 2\tilde{\mathbf{x}}_t^T\mathbf{w} + |\tilde{\mathbf{x}}_t|^2 \leq |\mathbf{w}|^2 + |\tilde{\mathbf{x}}_t|^2 \leq |\mathbf{w}|^2 + (R+\alpha)^2.$$

To understand the middle inequality note that by Lemma 13, when a mistake is made on a positive example $\mathbf{x}_t$, $\tilde{\mathbf{x}}_t^T\mathbf{w} \leq 0$. The last inequality comes from $R = \max_t |\mathbf{z}_t|$ implies $\max_t |\tilde{\mathbf{x}}_t| \leq R + \alpha$.

Similarly, if we make a mistake on a negative example $\mathbf{x}_t$, then we have:

$$(\mathbf{w}-\tilde{\mathbf{x}}_t)^T(\mathbf{w}-\tilde{\mathbf{x}}_t) = |\mathbf{w}|^2 - 2\tilde{\mathbf{x}}_t^T\mathbf{w} + |\tilde{\mathbf{x}}_t|^2 \leq |\mathbf{w}|^2 + |\tilde{\mathbf{x}}_t|^2 \leq |\mathbf{w}|^2 + (R+\alpha)^2.$$

By Lemma 13, when a mistake is made on a negative example $\mathbf{x}_t$, $\tilde{\mathbf{x}}_t^T\mathbf{w} \geq 0$, which implies the middle inequality.

Finally, if the algorithm makes M mistakes, then $\mathbf{w}^T\mathbf{w}^* \geq M$ and $|\mathbf{w}|^2 \leq M(R+\alpha)^2$, or equivalently, $|\mathbf{w}| \leq (R+\alpha)\sqrt{M}$. Using the fact that $\mathbf{w}^T\mathbf{w}^*/|\mathbf{w}^*| \leq |\mathbf{w}|$, we have

$$M/|\mathbf{w}^*| \leq (R+\alpha)\sqrt{M} \implies \sqrt{M} \leq (R+\alpha)|\mathbf{w}^*| \implies M \leq (R+\alpha)^2|\mathbf{w}^*|^2.$$

□

5.4 Known Weighted ℓ_1 Costs

In this section, we provide an algorithm for the weighted ℓ_1 costs setting. Unlike the ℓ_2 case, the modifications to the classical Perceptron algorithm in Algorithm 9 do not suffice; and our algorithm for this setting is more involved. Here is the key difference: In the ℓ_2 costs setting, the individuals always manipulate in direction of the current classifier $\mathbf{w}$. However, in the weighted ℓ_1 setting this is no longer the case. This brings up two challenges to our approach. First, there may be multiple utility maximizing manipulation directions. Second, the manipulation direction may have a negative dot product with $\mathbf{w}^*$. We overcome these two challenges and provide an algorithm for this setting.

As a reminder, in the weighted ℓ_1 costs setting, there are coordinate unit vectors $\{\mathbf{e}_1, \cdots, \mathbf{e}_d\}$ with cost of manipulation $1/\alpha_i$ along $\mathbf{e}_i$. We need to make one further assumption for this setting. We assume for all $1 \leq i \leq d$, $\mathbf{e}_i^T \mathbf{w}^* \geq 0$. In other words, we assume that each feature is defined so that larger is better. This is natural for settings such as hiring, admissions, loan applications, etc.

Overview of Algorithm 10. The algorithm starts by predicting all points as positive until it makes a mistake. Note that during this period, individuals do not have incentive to manipulate. From that point on, the algorithm classifies all points $\mathbf{x}_t$ such that $\mathbf{x}_t^T\mathbf{w}/|\mathbf{w}| - \alpha_i\mathbf{w}^T\mathbf{e}_i/|\mathbf{w}| \geq 0$ as positive, and the rest as negative; where $\mathbf{e}_i$ is the manipulation direction which will be defined later. Similar to Algorithm 9, whenever the algorithm makes a mistake, the predictor $\mathbf{w}$ is updated with a surrogate value, $\tilde{\mathbf{x}}_t$, in Definition 6.

Compared to Algorithm 9, we have two further steps. As discussed above, the first challenge with weighted ℓ_1 costs is that with an arbitrary $\mathbf{w}$, there may be multiple utility maximizing manipulation directions, and we may not be able to distinguish along which vector individuals manipulated. Since in the weighted ℓ_1 costs setting, the cost of manipulation can be written as a convex combination of costs in coordinate vectors, there always exists a coordinate vector, $\mathbf{e}_i$, such that manipulating along that is utility maximizing. Consider all the coordinate vectors like $\mathbf{e}_j$ that are utility maximizing, i.e., have the highest $\alpha_j \cdot \mathbf{w}^T\mathbf{e}_j/|\mathbf{w}|$. To make the manipulation direction unique, we add a *tie-breaking step* to the algorithm. This step adds a small multiple $\eta > 0$, of an arbitrary utility maximization coordinate vector $\mathbf{e}_i$, to $\mathbf{w}$ to break the tie. Note that any positive value of η breaks the tie. We set this value in our analysis purposes in Theorem 11 in a way to make sure the number of

mistakes our algorithm makes does not increase much.

We need to add another step to address the second challenge: With an arbitrary $\mathbf{w}$ the direction that the individuals manipulate along may not have a positive dot product with $\mathbf{w}^*$, i.e., the individuals may choose to move along one of the vectors $\{-\mathbf{e}_1, \cdots, -\mathbf{e}_d\}$. In order to incentivize individuals to only manipulate along $\{\mathbf{e}_1, \cdots, \mathbf{e}_d\}$, and not $\{-\mathbf{e}_1, \cdots, -\mathbf{e}_d\}$, we do the following *correction step* after each update. If $\mathbf{e}_j^T\mathbf{w} < 0$ for any $\mathbf{e}_j \in \{\mathbf{e}_1, \cdots, \mathbf{e}_d\}$, we set the j^{th} coordinate of $\mathbf{w}$ to 0 by adding the smallest multiple of $\mathbf{e}_j$, denoted by μ_j, to $\mathbf{w}$ to make $\mathbf{e}_j^T\mathbf{w}$ nonnegative. Therefore, $\mu_j = 0$ if $\mathbf{e}_j^T\mathbf{w} \geq 0$, and $\mu_j = -\mathbf{e}_j^T\mathbf{w}$, otherwise; implying $\forall j\ \mu_j \geq 0$.

With the unique manipulation direction, similar to the ℓ_2 costs setting, we are now able to choose a surrogate value along the manipulation direction.

Definition 6 ($\tilde{\mathbf{x}}_t$, surrogate data point in weighted ℓ_1 setting)**.** *Let* $\mathbf{e}_i$ *be the* unique *utility maximizing coordinate vector, i.e.,* $i = \arg\max_j \alpha_j \mathbf{w}^T\mathbf{e}_j/|\mathbf{w}|$. *We define surrogate data point,* $\tilde{\mathbf{x}}_t$*, as follows.*

$$\tilde{\mathbf{x}}_t = \begin{cases} \mathbf{x}_t - \mathbf{e}_i \cdot \alpha_i, & \text{if } \mathbf{x}_t \text{ is } - \text{ and } \frac{\mathbf{x}_t^T\mathbf{w}}{|\mathbf{w}|} = \alpha_i \cdot \frac{\mathbf{w}^T\mathbf{e}_i}{|\mathbf{w}|}; \\ \mathbf{x}_t, & \text{otherwise.} \end{cases}$$

Lemma 14. $\mu_j \leq R + \alpha_j$.

Algorithm 10: Strategic Perceptron for weighted ℓ_1 costs

$\mathbf{w} \leftarrow \mathbf{0}$;
for $t = 1, 2, \cdots$ **do**
  Given example $\mathbf{x}_t$:
  if $|\mathbf{w}|$ *is* 0 **then**
    predict +;
    if *the prediction was a mistake* **then**
      $\mathbf{w} \leftarrow \mathbf{w} - \mathbf{x}_t$;
      `/* Correction Step */`
      for $j = 1, 2, \cdots, d$ **do**
        $\mathbf{w} \leftarrow \mathbf{w} + \mu_j \mathbf{e}_j$, where $\mu_j = \max(0, -\mathbf{e}_j^T \mathbf{w})$;
      `/* Tie-breaking Step */`
      $i \leftarrow \arg\max_j \alpha_j \cdot \frac{\mathbf{w}^T \mathbf{e}_j}{|\mathbf{w}|}$;
      $\mathbf{w} \leftarrow \mathbf{w} + \eta \mathbf{e}_i$;
  else
    predict $sgn(\frac{\mathbf{x}_t^T \mathbf{w}}{|\mathbf{w}|} - \alpha_i \cdot \frac{\mathbf{w}^T \mathbf{e}_i}{|\mathbf{w}|})$;
    if *the prediction was a mistake and* $\mathbf{x}_t$ *was* + **then** $\mathbf{w} \leftarrow \mathbf{w} + \tilde{\mathbf{x}}_t$;
    if *the prediction was a mistake and* $\mathbf{x}_t$ *was* − **then** $\mathbf{w} \leftarrow \mathbf{w} - \tilde{\mathbf{x}}_t$;
    `/* Correction Step */`
    for $j = 1, 2, \cdots, d$ **do**
      $\mathbf{w} \leftarrow \mathbf{w} + \mu_j \mathbf{e}_j$ where $\mu_j = \max(0, -\mathbf{e}_j^T \mathbf{w})$;
    `/* Tie-breaking Step */`
    $i \leftarrow \arg\max_j \alpha_j \cdot \frac{\mathbf{w}^T \mathbf{e}_j}{|\mathbf{w}|}$;
    $\mathbf{w} \leftarrow \mathbf{w} + \eta \mathbf{e}_i$;

Proof. We can show at the end of each round, $\mathbf{e}_j^T \mathbf{w} \geq 0$. Initially, $\mathbf{w} = 0$, therefore $\mathbf{e}_j^T \mathbf{w} = 0$. Suppose at the end of round $t-1$, $\mathbf{e}_j^T \mathbf{w} \geq 0$. Assume in round t, $\mathbf{w}$ gets updated by adding or subtracting $\tilde{\mathbf{x}}_t$ or $\mathbf{x}_t$. By assumption, the j^{th} coordinate of $\mathbf{x}_t$ is in $[-R, R]$, and therefore the j^{th} coordinate of $\tilde{\mathbf{x}}_t$ is in $[-R - \alpha_j, R + \alpha_j]$. Taken together, $\mu_j \leq R + \alpha_j$. Note that by adding $\eta \mathbf{e}_i$ to $\mathbf{w}$, $\mathbf{e}_j^T \mathbf{w}$ remains nonnegative. □

The following theorem upper bounds the number of mistakes made by Algorithm 10.

Theorem 11. *Consider a sequence of examples before manipulation* $\mathbf{z}_1, \mathbf{z}_2, \cdots$, *which are observed as* $\mathbf{x}_1, \mathbf{x}_2, \cdots$. *Consider vector* $\mathbf{w}^*$ *such that* $\mathbf{z}_t^T \mathbf{w}^* \geq 1$ *for positive examples, and* $\mathbf{z}_t^T \mathbf{w}^* \leq -1$ *for negative examples. Algorithm 10 makes at most* $(1 + (d+1)(R+\alpha)^2)|\mathbf{w}^*|^2$ *mistakes, where* $R = \max_t |\mathbf{z}_t|$, *and* $\alpha = \max\{\alpha_1, \cdots, \alpha_d\}$.

Proof. Similar to the proof of Theorem 10, we keep track of two quantities $\mathbf{w}^T \mathbf{w}^*$ and $|\mathbf{w}|^2$. First, we show each time a mistake is made, $\mathbf{w}^T \mathbf{w}^*$ increases by at least 1. Then we find an upper bound on the increase of $|\mathbf{w}|^2$.

Starting from the current $\mathbf{w}$, the algorithm follows three steps to update: addition/subtraction of $\tilde{\mathbf{x}}_t$, the correction step, and the tie-breaking step. As in the algorithm $\mathbf{e}_i$ is the manipulation direction.

If the algorithm makes a mistake on a positive example the new value of $\mathbf{w}$ is $\mathbf{w} + \tilde{\mathbf{x}}_t + \eta \mathbf{e}_i + \sum_j \mu_j \mathbf{e}_j$. Therefore,

$$\left(\mathbf{w} + \tilde{\mathbf{x}}_\mathbf{t} + \eta \mathbf{e}_i + \sum_j \mu_j \mathbf{e}_j\right)^T \mathbf{w}^* = \mathbf{w}^T \mathbf{w}^* + \tilde{\mathbf{x}}_t^T \mathbf{w}^* + \eta \mathbf{e}_i^T \mathbf{w}^* + \sum_j \mu_j \mathbf{e}_j^T \mathbf{w}^* \geq \mathbf{w}^T \mathbf{w}^* + 1;$$

where the inequality holds because first using the ideas from Lemma 12, $\tilde{\mathbf{x}}_t^T \mathbf{w}^* \geq 1$ for the positive examples the algorithm makes a mistake on and

$\tilde{\mathbf{x}}_t^T \mathbf{w}^* \leq -1$ for the negative examples the algorithm makes a mistake on, and second, for all j, $\mathbf{e}_j^T \mathbf{w}^* \geq 0$ by assumption, and $\mu_j \geq 0$.

Similarly, If the algorithm makes a mistake on a negative example, we have:

$$\left(\mathbf{w} - \tilde{\mathbf{x}}_t + \eta \mathbf{e}_i + \sum_j \mu_j \mathbf{e}_j\right)^T \mathbf{w}^* = \mathbf{w}^T \mathbf{w}^* - \tilde{\mathbf{x}}_t^T \mathbf{w}^* + \eta \mathbf{e}_i^T \mathbf{w}^* + \sum_j \mu_j \mathbf{e}_j^T \mathbf{w}^* \geq \mathbf{w}^T \mathbf{w}^* + 1.$$

Next, on each mistake we claim $|\mathbf{w}|^2$ increases by at most $(d+1)(R+$

$\alpha)^2 + 1$. If the algorithm makes a mistake on a positive example, we have:

$$
\begin{aligned}
&\left|\mathbf{w} + \tilde{\mathbf{x}}_t + \eta\mathbf{e}_i + \sum_j \mu_j \mathbf{e}_j\right|^2 \\
&= |\mathbf{w} + \tilde{\mathbf{x}}_t + \eta\mathbf{e}_i|^2 + \left|\sum_j \mu_j \mathbf{e}_j\right|^2 + 2\left(\sum_j \mu_j \mathbf{e}_j\right)^T (\mathbf{w} + \tilde{\mathbf{x}}_t + \eta\mathbf{e}_i) \\
&= |\mathbf{w} + \tilde{\mathbf{x}}_t + \eta\mathbf{e}_i|^2 + \sum_j |\mu_j \mathbf{e}_j|^2 + 2\sum_j \mu_j \mathbf{e}_j^T (\mathbf{w} + \tilde{\mathbf{x}}_\mathbf{t} + \eta\mathbf{e}_i) \\
&\leq |\mathbf{w} + \tilde{\mathbf{x}}_\mathbf{t} + \eta\mathbf{e}_i|^2 + \sum_j |\mu_j \mathbf{e}_j|^2 + 2\sum_j \eta\mu_j \mathbf{e}_j^T \mathbf{e}_i \\
&= |\mathbf{w} + \tilde{\mathbf{x}}_\mathbf{t} + \eta\mathbf{e}_i|^2 + \sum_j |\mu_j|^2 + 2\eta\mu_i \\
&= |\mathbf{w}|^2 + |\tilde{\mathbf{x}}_t|^2 + |\eta\mathbf{e}_i|^2 + 2\mathbf{w}^T\tilde{\mathbf{x}}_t + 2\eta\mathbf{w}^T\mathbf{e}_i + 2\eta\tilde{\mathbf{x}}_t^T\mathbf{e}_i + \sum_j |\mu_j|^2 + 2\eta\mu_i \\
&\leq |\mathbf{w}|^2 + (R+\alpha)^2 + \eta^2 + 0 + 2\eta|\mathbf{w}| + 2\eta(R+\alpha) + d(R+\alpha)^2 + 2\eta(R+\alpha) \\
&\leq |\mathbf{w}|^2 + (d+1)(R+\alpha)^2 + \eta^2 + \eta(2|\mathbf{w}| + 4(R+\alpha)) \\
&\leq |\mathbf{w}|^2 + (d+1)(R+\alpha)^2 + 1/4 + 1/2 \\
&\leq |\mathbf{w}|^2 + (d+1)(R+\alpha)^2 + 1;
\end{aligned}
$$

where the first equality is the result of expansion. The second uses $\mathbf{e}_j^T\mathbf{e}_k = 0$ for $j \neq k$. The inequality in the third row uses $\mu_j = 0$ when $\mathbf{e}_j^T(\mathbf{w} + \tilde{\mathbf{x}}_t) \geq 0$, and $\mu_j > 0$ when $\mathbf{e}_j^T(\mathbf{w} + \tilde{\mathbf{x}}_t) < 0$, implying $\mu\mathbf{e}_j^T(\mathbf{w} + \tilde{\mathbf{x}}_t) \leq 0$. The fourth row uses $\mathbf{e}_j^T\mathbf{e}_k = 0$ for $k \neq j$ and $\mathbf{e_j}^T\mathbf{e_j} = 1$. The fifth row is the result of expansion.

The sixth row substitutes each term with an upper bound using $|\tilde{\mathbf{x}}_t| \leq R + \alpha$ and $\mathbf{w}^T \tilde{\mathbf{x}}_t \leq 0$, similar to the arguments from Lemma 13, and $\mu_j \leq R + \alpha$, by Lemma 14. The eighth row results by setting $\eta = \frac{1}{4|\mathbf{w}|+8(R+\alpha)+2}$. The last row sums up and upper bounds similar terms.

Similarly, if the algorithm makes a mistake on a negative example, we have:

$$\left| \mathbf{w} - \tilde{\mathbf{x}}_t + \eta \mathbf{e}_i + \sum_j \mu_j \mathbf{e}_j \right|^2 \leq |\mathbf{w}|^2 + (d+1)(R+\alpha)^2 + 1.$$

Therefore, after each mistake, $|\mathbf{w}|^2$ increases by at most $(d+1)(R+\alpha)^2+1$. The rest of the proof is similar to the proof of Theorem 10, concluding that the total number of mistakes is at most $((d+1)(R+\alpha)^2+1)|\mathbf{w}^*|^2$. □

5.5 Unknown Costs

The main result of this section is generalizing our algorithms to the unknown costs setting. The generalization holds for ℓ_2 costs . However, it does not extend fully to weighted ℓ_1 costs and only works for a specific case. The algorithm for unknown ℓ_2 costs is presented in Section 5.5. The case of unknown ℓ_1 costs is studied in Section 5.5.

ℓ_2 *Costs*

In this section, we provide an algorithm that makes at most a bounded number of mistakes when the manipulation cost, $1/\alpha$, is unknown. Algorithm 9 is used as a subroutine to evaluate our estimate of α. First, we show Algorithm 9 works efficiently if the estimated value, α', is in proximity of the real value (when α' is in the interval of length $\gamma/2$ below α). Using this idea we can run a linear search for α with step size $\gamma/2$. However, we show we can do better than a linear search. The key ingredient that lets us outperform the linear search is the ability to distinguish whether the estimate is below or above the real value. Using this idea we run a binary search to find a proper estimate and come up with an efficient algorithm.

For convenience, we will present the algorithm assuming γ is known. At

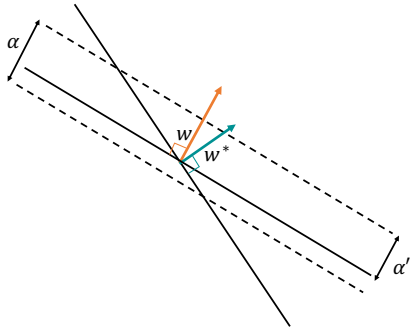


Figure 5.4: Strategic Perceptron with unknown manipulation cost, when $\alpha \geq \alpha'$. The top dashed line represents the manipulation hyperplane. The margin between the two dashed lines represents the forbidden region.

the end we show how to remove this assumption. Below, we explain these steps more formally.

Case 1: $0 \leq \alpha - \alpha' \leq \gamma/2$

First, we consider the case of $0 \leq \alpha - \alpha' \leq \gamma/2$. Suppose Algorithm 9 takes α' instead of α as input. Also, suppose $\tilde{\mathbf{x}}_t$ is defined with respect to α' instead of α. In Proposition 1, we show if $0 \leq \alpha - \alpha' \leq \gamma/2$, Algorithm 9 with these modifications, makes at most $4(R + \alpha' + \gamma/2)^2|\mathbf{w}^*|^2$ mistakes. We need the following two lemmas for proving the proposition. Proofs of Lemma 15 Lemma 16 are along the lines of proofs of Lemmas 12 and 13 respectively.

Lemma 15. *Consider data points $\tilde{\mathbf{x}}_t$ as defined in Definition 5 w.r.t. α' such that $0 \leq \alpha - \alpha' \leq \gamma/2$. These data points are $1/2$-separable; i.e., for positive data points, $\tilde{\mathbf{x}}_t^T\mathbf{w}^* \geq 1/2$; and for negative data points, $\tilde{\mathbf{x}}_t^T\mathbf{w}^* \leq -1/2$. Also, throughout the execution of Algorithm 9 with α', $\mathbf{w}^T\mathbf{w}^* \geq 0$.*

Proof. The proof uses the same three steps as Lemma 12. The first and the third steps are identical to Lemma 12. Here, we argue for the second step, i.e., if at the end of step $t-1$, $\mathbf{w}^T\mathbf{w}^* \geq 0$, then at step t, $\tilde{\mathbf{x}}_t^T\mathbf{w}^* \geq 1/2$ for positive points, and $\tilde{\mathbf{x}}_t^T\mathbf{w}^* \leq -1/2$ for negative points.

When Algorithm 9 is run with α', by Definition 5, for any points such that $\mathbf{x}_t^T\mathbf{w}/|\mathbf{w}| \neq \alpha'$, we have $\tilde{\mathbf{x}}_t = \mathbf{x}_t$. By Observation 1, these points are

not manipulated, i.e., $\mathbf{x}_t = \mathbf{z}_t$. This implies $\tilde{\mathbf{x}}_t = \mathbf{z}_t$ which implies the claim for these points. Thus, we only need to argue for the data points such that $\mathbf{x}_t^T \mathbf{w}/|\mathbf{w}| = \alpha'$. Consider such data points. For the positive data points, we have, $\tilde{\mathbf{x}}_t = \mathbf{x}_t = \mathbf{z}_t + \beta \cdot \mathbf{w}/|\mathbf{w}|$, where $0 \leq \beta \leq \alpha$. Therefore, $\tilde{\mathbf{x}}_t^T \mathbf{w}^* = \mathbf{z}_t^T \mathbf{w}^* + \beta \cdot \mathbf{w}^T \mathbf{w}^*/|\mathbf{w}| \geq \mathbf{z}_t^T \mathbf{w}^* \geq 1$. The first inequality holds because by the assumption of this step, $\mathbf{w}^T \mathbf{w}^* \geq 0$. On the other hand, for the negative data points we have $\tilde{\mathbf{x}}_t = \mathbf{x}_t - \alpha' \cdot \mathbf{w}/|\mathbf{w}|$, where $\mathbf{x}_t = \mathbf{z}_t + \beta \cdot \mathbf{w}/|\mathbf{w}|$ and $0 \leq \beta \leq \alpha$. This implies $\tilde{\mathbf{x}}_t = \mathbf{z}_t + (\beta - \alpha') \cdot \mathbf{w}/|\mathbf{w}|$. By multiplying with $\mathbf{w}^*$, we get $\tilde{\mathbf{x}}_t^T \mathbf{w}^* = \mathbf{z}_t^T \mathbf{w}^* + (\beta - \alpha') \cdot \mathbf{w}^T \mathbf{w}^*/|\mathbf{w}| \leq \mathbf{z}_t^T \mathbf{w}^* + (\alpha - \alpha') \cdot \mathbf{w}^T \mathbf{w}^*/|\mathbf{w}|$. Using $0 \leq \alpha - \alpha' \leq \gamma/2$ and $\gamma = 1/|\mathbf{w}^*|$, we have $\tilde{\mathbf{x}}_t^T \mathbf{w}^* \leq \mathbf{z}_t^T \mathbf{w}^* + \mathbf{w}^T \mathbf{w}^*/(2|\mathbf{w}^*||\mathbf{w}|) \leq \mathbf{z}_t^T \mathbf{w}^* + 1/2 \leq -1/2$. □

Lemma 16. *Suppose Algorithm 9 is run with α' such that $0 \leq \alpha - \alpha' \leq \gamma/2$. When the algorithm makes a mistake on a positive example $\mathbf{x}_t$, $\tilde{\mathbf{x}}_t^T \mathbf{w} \leq 0$; and when it makes a mistake on a negative example $\mathbf{x}_t$, $\tilde{\mathbf{x}}_t^T \mathbf{w} \geq 0$.*

Proof. First, we consider the positive points. The algorithm makes a mistake on a positive example only if $\mathbf{x}^T \mathbf{w}/|\mathbf{w}| < \alpha'$. Similar to Observation 2, in this case there is a margin without any observed data points. However, as illustrated in Figure 5.4, this margin is located differently; such that for no points, $\alpha' - \alpha < \mathbf{x}^T \mathbf{w}/|\mathbf{w}| < \alpha'$. Thus, for any positive example that the algorithm makes a mistake on, $\mathbf{x}^T \mathbf{w} \leq 0$. By Definition 5, $\tilde{\mathbf{x}}_t = \mathbf{x}_t$ for all

positive examples. Therefore, $\mathbf{x}^T\mathbf{w} \leq 0$ implies $\tilde{\mathbf{x}}^T\mathbf{w} \leq 0$. Second, we consider negative points. For negative examples, the algorithm makes a mistake only if $\mathbf{x}^T\mathbf{w}/|\mathbf{w}| \geq \alpha'$. If the inequality is strict, i.e., $\mathbf{x}^T\mathbf{w}/|\mathbf{w}| > \alpha'$, by Definition 5, $\tilde{\mathbf{x}}_t = \mathbf{x}_t$, and therefore $\tilde{\mathbf{x}}_t^T\mathbf{w} \geq 0$. If $\mathbf{x}^T\mathbf{w}/|\mathbf{w}| = \alpha'$, again using Definition 5, we have $\tilde{\mathbf{x}}^T\mathbf{w} = 0$. □

Proposition 1. *When* $0 \leq \alpha - \alpha' \leq \gamma/2$*, Algorithm 9 makes at most* $4(R + \alpha' + \gamma/2)^2|\mathbf{w}^*|^2$ *mistakes.*

Proof. Using Lemmas 15 and 16, the rest of the proof is similar to Theorem 10 and is deferred to the Appendix. □

Case 2: $\alpha < \alpha'$

Suppose α' is larger than α. By Observation 2, when Algorithm 9 is run with the real value of α, no data point is observed by algorithm in the margin $0 < \mathbf{x}_t^T\mathbf{w}/|\mathbf{w}| < \alpha$. However, when the estimate is larger, since we overestimate by how far individuals can manipulate, Observation 2 no longer holds. Therefore, if the algorithm observes a point in the margin $0 < \mathbf{x}_t^T\mathbf{w}/|\mathbf{w}| < \alpha'$, we realize that the estimate is large, and we need to refine it. On the other hand, while we have not observed any such points, the algorithm makes at most $(R + \alpha')^2|\mathbf{w}^*|^2$ mistakes. This statement is summarized and proved below.

Proposition 2. *Suppose Algorithm 9 is run with α', such that $\alpha' > \alpha$, and is halted if for a data-point $\mathbf{x}_t$, $0 < \mathbf{x}_t^T \mathbf{w}/|\mathbf{w}| < \alpha'$. This modified algorithm makes at most $(R+\alpha')^2|\mathbf{w}^*|^2 + 1$ mistakes.*

Proof. Similar to the proof of Theorem 10, the maximum number of mistakes Algorithm 9 with estimated manipulation cost $1/\alpha'$ makes on observed data points $\mathbf{x}_t$ where $\mathbf{x}_t^T \mathbf{w}/|\mathbf{w}| \leq 0$ or $\mathbf{x}_t^T \mathbf{w}/|\mathbf{w}| \geq \alpha'$ is at most $(R+\alpha')^2|\mathbf{w}^*|^2$. If a data point $\mathbf{x}_t$ is observed such that $0 < \mathbf{x}_t^T \mathbf{w}/|\mathbf{w}| < \alpha'$, it implies $\alpha' > \alpha$ and the algorithm halts, and at most one more mistake is made on this data point. Therefore, the total number of mistakes is at most $(R+\alpha')^2|\mathbf{w}^*|^2 + 1$. □

Case 3: $\alpha' < \alpha - \gamma/2$

We infer from Propositions 1 and 2 that if the number of mistakes is greater than $\max\{4(R+\alpha'+\gamma/2)^2|\mathbf{w}^*|^2, (R+\alpha')^2|\mathbf{w}^*|^2 + 1\} = 4(R+\alpha'+\gamma/2)^2|\mathbf{w}^*|^2$ then $\alpha' < \alpha - \gamma/2$. Note that the equality holds since the number of mistakes is an integer.

Putting Everything Together

After discussing the three cases, we are now ready to explain Algorithm 11. This algorithm uses a binary search scheme to find a predictor in a bounded number of mistakes. The algorithm starts with $\alpha' = 0$. For each

Algorithm 11: Strategic Perceptron with unknown manipulation cost

$\alpha'' \leftarrow 0, \alpha' \leftarrow 0$;
while *examples are arriving* **do**
  Run Algorithm 9 with estimate α' on the sequence of arriving examples, halt if $\#mistakes > 4(R + \alpha' + \gamma/2)^2|\mathbf{w}^*|^2$ or if for an example $\mathbf{x}_t$, $0 < \frac{\mathbf{x}_t^T \mathbf{w}}{|\mathbf{w}|} < \alpha'$;
  if $\#mistakes > 4(R + \alpha' + \gamma/2)^2|\mathbf{w}^*|^2$ **then**
    `/* guessed value` α' `is small. */`
    $\alpha'' \leftarrow \alpha'$;
    $\alpha' \leftarrow \min\{\max\{2\alpha', \gamma/2\}, R\}$;
    continue;
  else if *for an example* $\mathbf{x}_t$, $0 < \frac{\mathbf{x}_t^T \mathbf{w}}{|\mathbf{w}|} < \alpha'$ **then**
    `/* guessed value` α' `is large. */`
    $\alpha' \leftarrow (\alpha'' + \alpha')/2$;
    continue;

fixed α' we consider $4(R+\alpha'+\gamma/2)^2|\mathbf{w}^*|^2$ as the maximum number of allowed mistakes. Whenever we exceed this bound using the discussion in Section 5.5 we learn that α' is too small. Also whenever we see a data point $\mathbf{x}_t$ such that $0 \leq \mathbf{x}_t^T \mathbf{w}/|\mathbf{w}| < \alpha$ as explained above we learn that α' is too large. Distinguishing between the cases where α' is too large or too small allows us to refine the upper bound and lower bound on α' until $0 \leq \alpha - \alpha' \leq \gamma/2$. The following theorem shows that the total number of mistakes is bounded during the whole process.

Theorem 12. *Algorithm 11 makes at most* $\mathcal{O}(R^2|\mathbf{w}^*|^2 \log(R|\mathbf{w}^*|))$ *mistakes.*

Proof. In Algorithm 11, the candidates for α are $\gamma/2$ apart and the number of them is $2R|\mathbf{w}^*|$. Since we are doing a binary search on these candidates, the to-

tal number of iterations of binary search is at most $\log(2R|\mathbf{w}^*|)$. Proposition 1, Proposition 2, and Theorem 10, show that in each iteration the total number of mistakes is bounded by $\max\{4(R+\alpha'+\gamma/2)^2|\mathbf{w}^*|^2, (R+\alpha')^2|\mathbf{w}^*|^2+1\}$. Since we are assuming $\alpha' \leq R$, the total number of mistakes is at most $\mathcal{O}(R^2|\mathbf{w}^*|^2 \cdot \log(R|\mathbf{w}^*|))$ and the proof is complete. □

Unknown γ

In the previous steps we assumed knowledge of γ. However, this assumption is not necessary and we can remove it in the following way. Starting from a guess of $|\mathbf{w}^*| = \frac{1}{2R}$ (i.e., a guess of $\gamma = 2R$), repeat the following procedure: for each guessed value of $|\mathbf{w}^*|$, Algorithm 11 is executed and if it makes more than the mistake bound of $\mathcal{O}(R^2|\mathbf{w}^*|^2 \log(R|\mathbf{w}^*|))$, the guessed value for $|\mathbf{w}^*|$ is doubled (i.e., the guessed value of γ is halved) and the procedure is repeated. We show by putting this wrapper around Algorithm 11, the total number of mistakes remains in the same order of magnitude:

$$\sum_{i=-1}^{\log 2R|\mathbf{w}^*|} R^2 \left(\frac{|\mathbf{w}^*|}{2^i}\right)^2 \log\left(\frac{R|\mathbf{w}^*|}{2^i}\right) = \mathcal{O}(R^2|\mathbf{w}^*|^2 \log(R|\mathbf{w}^*|))$$

Weighted ℓ_1 Costs

As observed in Section 5.4, in order for the strategic Perceptron algorithm to work in the weighted ℓ_1 costs model, it is necessary to identify in what direction the individuals manipulate. The tie-breaking step in Algorithm 10, ensured that the manipulation direction is unique and identifiable. In the unknown costs model, we need to make a guess for the cost in each direction. Since the guessed values are not accurate, we no longer can use them for a tie-breaking step and determine the manipulation direction. This restrains us from having an efficient algorithm for the general case of ℓ_1 costs. However, for a special case where manipulation is possible in a single direction (finite cost in direction $\mathbf{e}_1$ and infinite in the others), the manipulation direction is known and the ideas of Algorithm 11 extend to this case.

5.6 Different Costs

In the previous sections, we assumed all individuals have the same utility function. In this section, we show this assumption is not critical for our result and our algorithms still make a bounded number of mistakes and perform almost as well as long as the utility functions are close enough.

More particularly, suppose in the ℓ_2 costs setting, at each time t, the amount that an individual can move, α_t, is upper bounded by $\alpha_{\max}$ and lower bounded by $\alpha_{\min}$ such that $0 \leq \alpha_{\max} - \alpha_{\min} \leq \gamma/2$. Using the ideas presented in Section 5.5, we can show that running Algorithm 9 with $\alpha_{\min}$ as the input and the surrogate data points $\tilde{\mathbf{x}}_t$ defined with respect to $\alpha_{\min}$ makes a bounded number of mistakes.

Corollary 3. *Suppose for all t, $\alpha_{\min} \leq \alpha_t \leq \alpha_{\min} + \gamma/2$. Algorithm 9 by using parameter $\alpha_{\min}$ makes at most $4(R + \alpha_{\min} + \gamma/2)|\mathbf{w}^*|^2$ number of mistakes.*

Proof Outline. In Section 5.5, the guessed value of α that is used as the input to Algorithm 9, is at most $\gamma/2$ smaller than the real value. Similarly, in this case, $\alpha_{\min}$ is at most $\gamma/2$ smaller than any α_t. With a small difference in the terminology of the proofs of Lemmas 15 and 16, their statements hold and Proposition 1 directly implies this corollary. □

Weighted ℓ_1 Costs

Due to similar reasons explained in Section 5.5, the previous result does not extend to general case of weighted ℓ_1 costs but extends to the special case where manipulation is possible in a single direction.

5.7 Target Classifier Not Crossing the Origin

In this section, we propose an algorithm for the setting where unmanipulated data points are linearly separable, however not by a linear separator passing through the origin. Ordinarily (in the non-strategic setting), this would be handled by creating an extra fake coordinate, giving each example a value of 1 in that coordinate, and thereby reducing to the case where the separator crosses the origin, as explained in Extension 1. However, in the strategic setting, this reduction breaks down because the condition that $\mathbf{w}^T\mathbf{w}^* \geq 0$ (given in Lemma 12) is no longer sufficient to guarantee the quality of $\tilde{\mathbf{x}}_t$, since agents cannot manipulate in this new coordinate (they are no longer manipulating in the direction of $\mathbf{w}$). Instead, we present a different reduction here that is robust to strategic behavior.

For the case of ℓ_2 costs, we provide Algorithm 12 and show that the number of mistakes it makes is at most $\mathcal{O}(R^3|\mathbf{w}^*|^3)$.

Overview of Algorithm 12. Assume the unmanipulated data points are separable by a linear separator $\mathbf{w}^{*T}\mathbf{z} + b = 0$. Suppose we can find an arbitrary point $\mathbf{p}^*$ such that $\mathbf{w}^{*T}\mathbf{p}^* + b = 0$. If we set the point $\mathbf{p}^*$ as the new origin, i.e.,

Algorithm 12: Strategic Perceptron with bias for ℓ_2 costs

$\mathbf{x}^+, \mathbf{x}^- \leftarrow \mathbf{0}, \mathbf{0}$;
while *examples are arriving* **do**
  predict $+$;
  if *the prediction was a mistake on a current example* $\mathbf{x_t}$ **then**
    $\mathbf{x}^- \leftarrow \mathbf{x}_t$;
    break;
while *examples are arriving* **do**
  predict $-$;
  if *the prediction was a mistake on a current example* $\mathbf{x_t}$ **then**
    $\mathbf{x}^+ \leftarrow \mathbf{x}_t$;
    break;
$\lambda \leftarrow 0$;
while *examples are arriving* **do**
  mistakes $\leftarrow 0$;
  `/* choose a point` $\mathbf{p}$ `on the line segment between` $\mathbf{x}^+$ `and` $\mathbf{x}^-$`. */`
  $\mathbf{p} \leftarrow (1-\lambda)\mathbf{x}^- + \lambda\mathbf{x}^+$;
  `/* set the origin to point` $\mathbf{p}$`. */`
  Run Algorithm 9 on the sequence of arriving examples in the new coordinate system, i.e. replace each example $\mathbf{x}_t$ with $\mathbf{x_t} - \mathbf{p}$, and halt if $mistakes > 4(2R+\alpha)^2|\mathbf{w}^*|^2$;
  `/*` $\mathbf{p}$ `is not close enough to` $\mathbf{p}^*$`, i.e.` $|\mathbf{p}-\mathbf{p}^*| > \gamma/2$`. Try a different` $\mathbf{p}$`. */`
  $\lambda \leftarrow \lambda + \gamma/2$;
  continue;

replacing each example $\mathbf{z}_t$ with $\mathbf{z}_t - \mathbf{p}^*$, then in the new coordinate system the unmanipulated data points are linearly separable by a separator that crosses the new origin, and we can use our previous algorithms. However, we are not able to necessarily find a point $\mathbf{p}^*$ such that $\mathbf{w}^{*T}\mathbf{p}^* + b = 0$. Instead, we show how to find a point $\mathbf{p}$ that is close enough to $\mathbf{p}^*$.

Initially, we find an unmanipulated positive example $\mathbf{x}^+$, and an unma-

nipulated negative example $\mathbf{x}^-$ by starting our algorithm in the following way: First, predict positive until the first mistake on a negative example $\mathbf{x}^-$ is made. Next, predict negative until the next mistake is made on some positive example $\mathbf{x}^+$. Consider the line segment between $\mathbf{x}^-$ and $\mathbf{x}^+$. There exists a point $\mathbf{p}^*$ on this line segment where $\mathbf{w}^{*T}\mathbf{p}^* + b = 0$. Consider a series of points on this line segment at distance $\gamma/2$ apart. For one of these points, which we call $\mathbf{p}$, $|\mathbf{p} - \mathbf{p}^*| \leq \gamma/2$. Lemma 17 shows if the origin is set to $\mathbf{p}$, i.e. each data point $\mathbf{z}_t$ is replaced with $\mathbf{z}_t - \mathbf{p}$, there exists a line passing through the new origin $\mathbf{p}$ that separates original data points with a margin of $\gamma/2$, meaning that for each unmanipulated negative example $\mathbf{z}_t$, $(\mathbf{z}_t - \mathbf{p})^T\mathbf{w}^* \geq 1/2$, and for each unmanipulated negative example $\mathbf{z}_t$, $(\mathbf{z}_t - \mathbf{p})^T\mathbf{w}^* \leq -1/2$. When the origin is set to $\mathbf{p}$, Lemma 18 shows that if Algorithm 9 is executed on the arrived examples in the new coordinate system, i.e. replacing each observed example $\mathbf{x}_t$ with $\mathbf{x_t} - \mathbf{p}$, the number of mistakes is at most $4(2R + \alpha)^2|\mathbf{w}^*|^2$. Putting all together, we propose Algorithm 12 that is a generalization of Algorithm 9 for the case that original examples are separable by a linear classifier with non-zero bias. Theorem 13 shows Algorithm 12 makes at most $\mathcal{O}(R^3|\mathbf{w}^*|^3)$ mistakes.

Lemma 17. *Assume the points* $\mathbf{z}_t$ *are separable by a linear separator* $\mathbf{w}^{*T}\mathbf{z} + b = 0$ *of margin* $\gamma = 1/|\mathbf{w}^*|$*, and let* $\mathbf{p}^*$ *be a point such that* $\mathbf{w}^{*T}\mathbf{p}^* + b = 0$.

Then, if $|\mathbf{p}^* - \mathbf{p}| \leq \gamma/2$*, the decision boundary* $(\mathbf{z} - \mathbf{p})^T \mathbf{w}^*$ *has a margin of separation* $\gamma/2$*.*

Proof. First, $|\mathbf{w}^{*T}\mathbf{p} - \mathbf{w}^{*T}\mathbf{p}^*| \leq 1/2$ because $|\mathbf{w}^{*T}\mathbf{p} - \mathbf{w}^{*T}\mathbf{p}^*| \leq |\mathbf{w}^*||\mathbf{p} - \mathbf{p}^*| \leq |\mathbf{w}^*|/2|\mathbf{w}^*| = 1/2$. So, for a positive data point $\mathbf{z}_t$, if $(\mathbf{z}_t - \mathbf{p}^*)^T \mathbf{w}^* \geq 1$, then $(\mathbf{z}_t - \mathbf{p})^T \mathbf{w}^* \geq 1/2$. Similarly, for a negative data point $\mathbf{z}_t$, if $(\mathbf{z}_t - \mathbf{p}^*)^T \mathbf{w}^* \leq -1$ then $(\mathbf{z}_t - \mathbf{p})^T \mathbf{w}^* \leq -1/2$. □

Lemma 18. *For a fixed guess* $\mathbf{p}$ *where* $|\mathbf{p}^* - \mathbf{p}| \leq \gamma/2$*, when the origin is set to* $\mathbf{p}$ *(i.e., each example* $\mathbf{x}$ *is replaced by* $\mathbf{x} - \mathbf{p}$*), then Algorithm 12 makes at most* $4(2R + \alpha)^2 |\mathbf{w}^*|^2$ *mistakes.*

Proof. Proof of this lemma is in the same lines as the proof of Theorem 10 with some modifications. First, Lemma 17 shows there exists a separator with margin of separation $\gamma/2$ passing through $\mathbf{p}$. By following the steps in Lemma 12, and using a margin of separation $\gamma/2$ instead of γ, we can show for any positive data point $\mathbf{x}_t$, $\tilde{\mathbf{x}}_t^T \mathbf{w}^* \geq 1/2$, and for any negative data point $\mathbf{x}_t$, $\tilde{\mathbf{x}}_t^T \mathbf{w}^* \leq -1/2$. Next, since each data point $\mathbf{x}_t$ is moved to $\mathbf{x}_t - \mathbf{p}$, $\max_t |\tilde{\mathbf{x}}_t| \leq 2R + \alpha$. Finally, by applying these bounds and following the steps in the proof of Theorem 10, the claim is proved. □

Theorem 13. *Algorithm 12 makes at most* $16R(2R + \alpha)^2 |\mathbf{w}^*|^3$ *mistakes in total.*

Proof. Since the length of the line segment between $\mathbf{x}^+$ and $\mathbf{x}^-$ is at most $2R$, and guesses tried on this line segment are $\gamma/2$ apart, Algorithm 12 tries at most $4R/\gamma$ guesses for $\mathbf{p}$ in total. For each guess, if the number of mistakes is greater than $4(2R+\alpha)^2|\mathbf{w}^*|^2$, the next guess is tried. By Lemma 18, if for a guess $\mathbf{p}$, $|\mathbf{p}-\mathbf{p}^*| \leq \gamma/2$, on all the examples that will arrive the number of mistakes does not exceed $4(2R+\alpha)^2|\mathbf{w}^*|^2$. Therefore the claim is proved. □

Weighted ℓ_1 Costs

We show a reduction from this setting to the case where unmanipulated examples are separable by a hyperplane passing through the origin. First, similar to the ℓ_2 case, an extra fake coordinate with a value of 1 is added to each example. The bias term b is absorbed into $\mathbf{w}^*$, i.e. replacing $\mathbf{w}^*$ with $(\mathbf{w}^*, b)$. Now for the positive examples $\mathbf{x}_t^T\mathbf{w}^* \geq 1$, and for the negative examples $\mathbf{x}_t^T\mathbf{w}^* \leq -1$. Since agents cannot manipulate in the direction of the fake coordinate, the cost of manipulation along this direction is set to be infinity. Next, we bound the number of mistakes that Algorithm 10 makes in this setting. Since a fake coordinate is added to each example, R increases by a value of at most 1. Since the bias term is absorbed into the $\mathbf{w}^*$, the value of $|\mathbf{w}|$ increases by at most b. As a result, Algorithm 10 makes at most $(1+(d+1)(R+\alpha+1)^2)(|\mathbf{w}^*|+b)^2$ number of mistakes. At the high level, the

reason the direct reduction goes through for the weighted ℓ_1 case but not the ℓ_2 case is that in the ℓ_1 case, agents only manipulate in coordinate directions, and we have assumed that $\mathbf{w}^*$ is non-negative in each coordinate direction. So, the algorithm is not hurt if in computing $\tilde{\mathbf{x}}_t$ it overestimates the amount by which the agent has manipulated. This is the property that breaks down in the ℓ_2 case.

5.8 Conclusions and Open Problems

In this work, we showed that if agents have the ability to manipulate their features within an ℓ_2 ball or a weighted ℓ_1 ball in order to be classified as positive, then the classic Perceptron algorithm may fail to achieve a bounded number of mistakes even when a perfect linear classifier exists. We then developed new Perceptron-style algorithms that achieve a finite mistake-bound, not much greater than the classic Perceptron bound in the non-strategic case, in both the ℓ_2 and weighted ℓ_1 manipulation setting. In the case that the manipulation costs are unknown to the learner—i.e., the radius of the ball in which agents can modify their features (or the per-coordinate radius in the weighted ℓ_1 case)—we provide an algorithm for the ℓ_2 costs setting and a specific case of the weighted ℓ_1 costs setting.

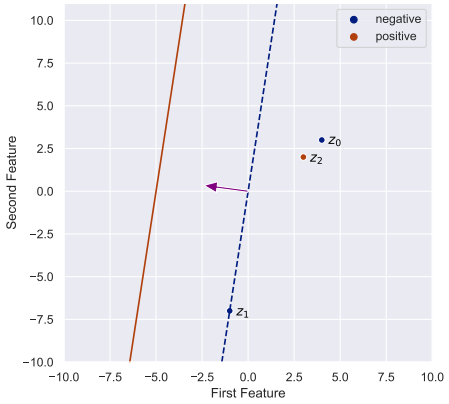
Our work suggests several open problems. First, designing an algorithm for the general case of weighted ℓ_1 costs when the costs of manipulation along each coordinate is unknown. This is challenging because given an observed data point, the learner doesn't know which direction it may have manipulated from, and this direction will change as the hypothesis classifier changes.

Second, for the case of inseparable data points, getting a bound in terms of the hinge-loss of the best separator with respect to the original data points $\mathbf{z}_1, \mathbf{z}_2, \cdots$. Our ideas in Section 5.3 can be extended to get a bound in terms of the hinge-loss of the best separator of surrogate data points $\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \cdots$. However, the more interesting question of getting a bound in terms of unmanipulated data points remains open. In the following, we show an example where Algorithm 9 makes an unbounded number of mistakes when data points are not perfectly separable, even though there exists a separator with bounded hinge-loss.

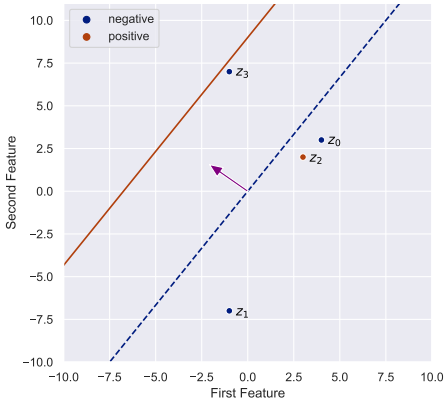
Example 3. *Consider data points* $\mathbf{z}_0 = (4, 3)$, $\mathbf{z}_1 = (-1, -7)$, $\mathbf{z}_2 = (3, 2)$, $\mathbf{z}_3 = (-1, 7)$, *and* $\mathbf{z}_4 = (3, -2)$ *arriving in order; and then the examples* $\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4$ *repeat forever. Examples* $\mathbf{z}_2$ *and* $\mathbf{z}_4$ *have positive labels, and* $\mathbf{z}_0$, $\mathbf{z}_1$ *and* $\mathbf{z}_3$ *have negative labels. Suppose that* $\alpha = 5$. *Note that there exists a vertical linear separator that only makes a mistake on* $\mathbf{z}_0$. *However, as shown below, Algorithm 9 will make an unbounded number of mistakes.*

Specifically, after arrival of $\mathbf{z}_0$*, we have* $\mathbf{w} = (-4, -3)$*. Next,* $\mathbf{z}_1$ *arrives, and since* $\mathbf{w}^T\mathbf{z}_1/|\mathbf{w}| = \alpha$*, the algorithm makes a mistake on* $\mathbf{z}_1$ *and classifies it as positive.* $\mathbf{z}_1$ *doesn't even have to manipulate, i.e.* $\mathbf{z}_1 = \mathbf{x}_1$*. Surrogate data point* $\tilde{\mathbf{x}}_1 = (3, -4)$ *is created, and is subtracted from* $\mathbf{w}$ *to get* $\mathbf{w} = (-7, 1)$*. Example* $\mathbf{z}_2$ *arrives and since* $\mathbf{w}^T\mathbf{z}_2 < 0$*, manipulation does not help, therefore,* $\mathbf{x}_2 = \mathbf{z}_2$*. Example* $\mathbf{z}_2$ *gets classified as negative mistakenly as shown in Figure 5.5(a). Also,* $\tilde{\mathbf{x}}_2 = \mathbf{x}_2$*. Since a mistake is made,* $\mathbf{w}$ *gets updated to* $\mathbf{w} + \tilde{\mathbf{x}}_2 = (-4, 3)$*. Next, negative example* $\mathbf{z}_3$ *arrives and since* $\mathbf{w}^T\mathbf{z}_3/|\mathbf{w}| = \alpha$*, it gets misclassified as positive as shown in Figure 5.5(b), and* $\mathbf{x}_3 = \mathbf{z}_3$*. Surrogate data point* $\tilde{\mathbf{x}}_\mathbf{3} = (3, 4)$ *is created and* $\mathbf{w}$ *is updated to* $\mathbf{w} - \tilde{\mathbf{x}}_\mathbf{3} = (-7, -1)$*. Next, positive example* $\mathbf{z}_4$ *arrives, and since* $\mathbf{w}^T\mathbf{z}_4 < 0$*, it does not manipulate and* $\mathbf{z}_4 = \mathbf{x}_4$*. It gets classified as negative mistakenly as shown in Figure 5.5(c). Also,* $\tilde{\mathbf{x}}_4 = \mathbf{x}_4$*.* $\mathbf{w}$ *is updated to* $\mathbf{w} + \tilde{\mathbf{x}}_4 = (-4, -3)$*. If the same four examples arrive over and over again, Algorithm 9 makes an unbounded number of mistakes. However, there exists a linear classifier* $\mathbf{w}^* = (1, 0)$ *which makes only one mistake in this scenario as shown in Figure 5.5(d).*

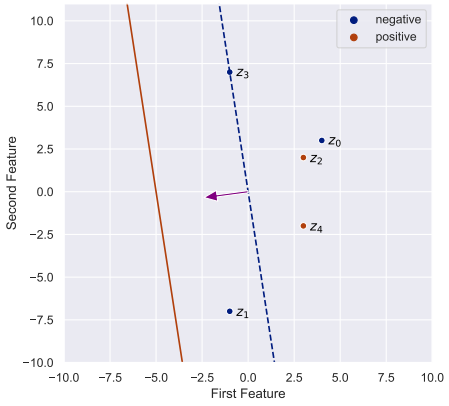
Third, when the separator of the original data points $\mathbf{z}_1, \mathbf{z}_2, \cdots$ does not cross the origin, as studied in Section 5.7, Algorithm 12 makes at most $O(R^3|\mathbf{w}^*|^3)$ mistakes in the ℓ_2 costs setting. Our last open problem is whether it is possible to improve the number of mistakes to $O(R^2|\mathbf{w}^*|^2)$.



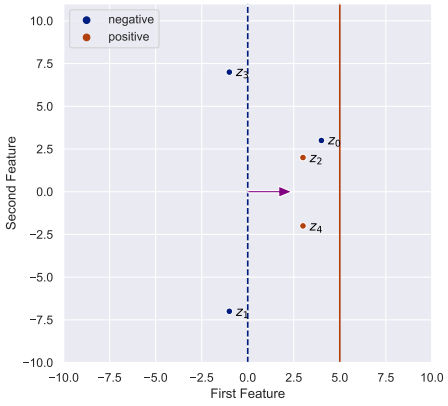
(a) The algorithm makes a mistake on $\mathbf{z}_2 = (3, 2)$ when $\mathbf{w} = (-7, 1)$.



(b) $\mathbf{z}_3 = (-1, 7)$ is mistakenly classified when $\mathbf{w} = (-4, 3)$.



(c) Positive example $\mathbf{z}_4 = (3, -2)$ is misclassified when $\mathbf{w} = (-7, -1)$.



(d) Data is not linearly separable, however, with $\mathbf{w}^* = (1, 0)$, only one mistake is made.

Figure 5.5: Example 3 shows Algorithm 9 makes an unbounded number of mistakes, where there is a separator with bounded hinge-loss. The dotted line in each figure shows the current $\mathbf{w}$, and the solid line shows the current classifier. The arrows show the positive direction of each classifier.

5.9 Authors

This Chapter was written by Saba Ahmadi, Hedyeh Beyhaghi, Avrim Blum, and Keziah Naggita. It is under submission to the Economics and Computation Conference (EC) 2021 [89].

6 Conclusion

In this thesis, we studied some social impacts of algorithms including diversity, fairness, and resilience to strategic behavior.

In Chapter 2, we generalized the notion of diverse b-matchings introduced by Ahmed et al [28]. In their model, they considered the problem of forming a bipartite b-matching from a set of workers to a set of teams, where the goal is to find a low-cost matching while forming diverse teams. In their model, they assume each worker has one feature. We generalized their model, by allowing each worker to have a set of features and designed a pseudo-polynomial time algorithm for this problem. Our algorithm is based on a negative cycle detection approach. Additionally, we proved that the problem of finding a diverse b-matching is NP-hard when the number of features is part of the input. In particular, this problem can be solved in polynomial time when each worker has exactly one feature. In terms of future direction, it would be interesting to design fast approximation algorithms for the problem

of diverse b-matching when the workers have multiple features. Extending our notion of diversity to the case of online matchings is another open question.

In Chapters 3, 4, we studied two different notions of fairness for the *correlation clustering* problem. In Chapter 3, we considered a notion of *group fairness* in correlation clustering where the goal is to generate clusters with minimum disagreements, where the distribution of a feature (e.g. gender) in each cluster is the same as the global distribution. Our guarantees are for the case of complete graphs, we leave open the question of finding approximation algorithms for fair correlation clustering in general graphs. In Chapter 4, we considered a different notion of fairness for correlation clustering. We defined a cluster-wise objective function that asks to minimize the maximum number of disagreements of each cluster. This notion respects the quality of the formed clusters. Kalhan et al. [20] later improved our results presented in Chapter 4, and derived a $(2+\epsilon)$-approximation algorithm for the case of general graphs. For the vertex-wise objective introduced by Puleo and Milenkovic [15], the best-known approximation ratio on complete graphs is 5 by Kalhan et al. [20]. The best known lower bound for this problem is 4/3 by Chatziafratis et al. [90]. Closing the gap between the lower and upper bounds remains open.

In Chapter 5, we discussed how to learn a linear classifier in presence of strategic agents that desire to be classified as positive and that are able to

modify their position by a limited amount, making the classifier not be able to observe the true position of agents but rather a position where the agent pretends to be. We illustrate an example where the traditional Perceptron algorithm fails to find a classifier in the strategic scenario, making an unbounded number of mistakes even though a perfect large-margin linear classifier exists. We provided a Perceptron style algorithm which makes a bounded number of mistakes in presence of strategic agents in both ℓ_2 and weighted-ℓ_1 manipulation costs. Our algorithms work for the scenario where the un-manipulated data points are separable. The question of finding a classifier in presence of strategic agents where the un-manipulated data points are inseparable remains open. The other open question is designing an algorithm for the general case of weighted ℓ_1 costs when the costs of manipulation along each coordinate is unknown.

In conclusion, in addition to group fairness in clustering, it would be interesting to study individual fairness in clustering. In a recent work, Kleindessner et al. [91] propose a new notion of individual fairness in clustering. In their notion, every data point is on average closer to the points in its cluster, compared to the points in other clusters. They show the problem of deciding whether a data set has an individually fair clustering in general is NP-hard. For the special case of a data set lying on a real line, they propose a dynamic

programming approach to find an individually fair clustering. It remains open to extend their notion of individual fair clustering to a general metric space. In particular, it would be interesting to find bicriteria approximation algorithms that satisfy both group fairness and individual fairness approximately. Furthermore, exploring connections between fairness and diversity in team formation is another direction that could be investigated. One open question is how to generalize our framework for the diverse matching problem to create teams that satisfy group fairness with respect to some features, while maximizing diversity with respect to some other features.

Bibliography

[1] Danielle Keats Citron and Frank Pasquale. The scored society: Due process for automated predictions. *Wash. L. Rev.*, 89:1, 2014.

[2] Marianne Bertrand and Sendhil Mullainathan. Are emily and greg more employable than lakisha and jamal? a field experiment on labor market discrimination. *American economic review*, 94(4):991–1013, 2004.

[3] Anthony W Flores, Kristin Bechtel, and Christopher T Lowenkamp. False positives, false negatives, and false analyses: A rejoinder to machine bias: Theres software used across the country to predict future criminals. and its biased against blacks. 2016.

[4] Amit Datta, Michael Carl Tschantz, and Anupam Datta. Automated experiments on ad privacy settings: A tale of opacity, choice, and discrimination. *Proceedings on privacy enhancing technologies*, 2015(1):92–112, 2015.

[5] Faez Ahmed, John P. Dickerson, and Mark Fuge. Diverse weighted bipartite b-matching. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 35–41, 2017.

[6] Maryam Karimzadehgan and ChengXiang Zhai. Constrained multi-aspect expertise matching for committee review assignment. In *ACM Conference on Information and Knowledge Management (CIKM)*, pages 1697–1700, 2009.

[7] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, and Sergei Vassilvitskii. Fair clustering through fairlets. In I. Guyon, U. V. Luxburg, S. Bengio,

H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5029–5037. Curran Associates, Inc., 2017.

[8] Ioana O. Bercea, Samir Khuller, Clemens Rösner, Melanie Schmidt, Martin Groß, Aounon Kumar, and Daniel R. Schmidt. On the cost of essentially fair clusterings. In Dimitris Achlioptas and Laszlo A. Vegh, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2019*, Leibniz International Proceedings in Informatics, LIPIcs. Schloss Dagstuhl- Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing, September 2019. 22nd International Conference on Approximation Algorithms for Combinatorial Optimization Problems and 23rd International Conference on Randomization and Computation, APPROX/RANDOM 2019 ; Conference date: 20-09-2019 Through 22-09-2019.

[9] Suman Bera, Deeparnab Chakrabarty, Nicolas Flores, and Maryam Negahbani. Fair algorithms for clustering. In *Advances in Neural Information Processing Systems*, pages 4955–4966, 2019.

[10] Saba Ahmadi, Sainyam Galhotra, Barna Saha, and Roy Schwartz. Fair correlation clustering. *arXiv preprint arXiv:2002.03508*, 2020.

[11] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Machine Learning*, 56(1-3):89–113, 2004.

[12] Nate Veldt, David F Gleich, and Anthony Wirth. A correlation clustering framework for community detection. In *Proceedings of the 2018 World Wide Web Conference*, pages 439–448, 2018.

[13] Francesco Bonchi, David Garcia-Soriano, and Edo Liberty. Correlation clustering: from theory to practice. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1972–1972, 2014.

[14] Jack P Hou, Amin Emad, Gregory J Puleo, Jian Ma, and Olgica Milenkovic. A new correlation clustering method for cancer mutation analysis. *Bioinformatics*, 32(24):3717–3728, 2016.

[15] Gregory Puleo and Olgica Milenkovic. Correlation clustering and biclustering with locally bounded errors. In *International Conference on Machine Learning*, pages 869–877, 2016.

[16] Yizong Cheng and George M Church. Biclustering of expression data. In *Ismb*, volume 8, pages 93–103, 2000.

[17] Panagiotis Symeonidis, Alexandros Nanopoulos, Apostolos Papadopoulos, and Yannis Manolopoulos. Nearest-biclusters collaborative filtering with constant values. In *International Workshop on Knowledge Discovery on the Web*, pages 36–55. Springer, 2006.

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

[19] Moses Charikar, Neha Gupta, and Roy Schwartz. Local guarantees in graph cuts and clustering. In Friedrich Eisenbrand and Jochen Koenemann, editors, *Integer Programming and Combinatorial Optimization*, pages 136–147, Cham, 2017. Springer International Publishing.

[20] Sanchit Kalhan, Konstantin Makarychev, and Timothy Zhou. Improved algorithms for correlation clustering with local objectives. *arXiv preprint arXiv:1902.10829*, 2019.

[21] Alvin E Roth. The evolution of the labor market for medical interns and residents: a case study in game theory. *Journal of political Economy*, 92(6):991–1016, 1984.

[22] Joanna Drummond, Andrew Perrault, and Fahiem Bacchus. Sat is an effective and complete method for solving stable matching problems with couples. In *IJCAI*, pages 518–525, 2015.

[23] Ryoji Kurata, Naoto Hamada, Atsushi Iwasaki, and Makoto Yokoo. Controlled school choice with soft bounds and overlapping types. *Journal of Artificial Intelligence Research*, 58:153–184, 2017.

[24] Laurent Charlin and Richard Zemel. The toronto paper matching system: an automated paper-reviewer assignment system. 2013.

[25] Xiang Liu, Torsten Suel, and Nasir Memon. A robust model for paper reviewer assignment. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 25–32, New York, NY, USA, 2014. ACM.

[26] John P. Dickerson and Tuomas Sandholm. Futurematch: Combining human value judgments and machine learning to match in dynamic environments. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 622–628. AAAI Press, 2015.

[27] Dimitris Bertsimas, Theodore Papalexopoulos, Nikolaos Trichakis, Yuchen Wang, Ryutaro Hirose, and Parsia A Vagefi. Balancing efficiency and fairness in liver transplant access: tradeoff curves for the assessment of organ distribution policies. *Transplantation*, 2019.

[28] Faez Ahmed, John P. Dickerson, and Mark Fuge. Diverse weighted bipartite b-matching. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, pages 35–41. AAAI Press, 2017.

[29] M. Minoux. *Solving integer minimum cost flows with separable convex cost objective polynomially*, pages 237–239. Springer Berlin Heidelberg, Berlin, Heidelberg, 1986.

[30] Senjuti Basu Roy, Ioanna Lykourentzou, Saravanan Thirumuruganathan, Sihem Amer-Yahia, and Gautam Das. Task assignment optimization in knowledge-intensive crowdsourcing. *The VLDB JournalThe International Journal on Very Large Data Bases*, 24(4):467–491, 2015.

[31] Joel Ross, Lilly Irani, M Silberman, Andrew Zaldivar, and Bill Tomlinson. Who are the crowdworkers?: shifting demographics in mechanical turk. In *CHI'10 extended abstracts on Human factors in computing systems*, pages 2863–2872. ACM, 2010.

[32] Christian R Østergaard, Bram Timmermans, and Kari Kristinsson. Does a different view create something new? the effect of employee diversity on innovation. *Research Policy*, 40(3):500–509, 2011.

[33] Vivian Hunt, Dennis Layton, and Sara Prince. Diversity matters. *McKinsey & Company*, 1:15–29, 2015.

[34] Hui Lin and Jeff A Bilmes. Learning mixtures of submodular shells with application to document summarization. *arXiv preprint arXiv:1210.4871*, 2012.

[35] Jaime Carbonell and Jade Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 335–336. ACM, 1998.

[36] Alex Kulesza, Ben Taskar, et al. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286, 2012.

[37] Faez Ahmed and Mark Fuge. Ranking ideas for diversity and quality. *Journal of Mechanical Design*, 140(1):011101, 2018.

[38] Paul Gölz and Ariel D Procaccia. Migration as submodular optimization. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[39] Nawal Benabbou, Mithun Chakraborty, Xuan-Vinh Ho, Jakub Sliwinski, and Yair Zick. Diversity constraints in public housing allocation. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 973–981. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

[40] Jing Wu Lian, Nicholas Mattei, Renee Noble, and Toby Walsh. The conference paper assignment problem: Using order weighted averages to assign indivisible goods. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[41] Shipra Agrawal, Morteza Zadimoghaddam, and Vahab Mirrokni. Proportional allocation: Simple, distributed, and diverse matching with high entropy. In *International Conference on Machine Learning (ICML)*, pages 99–108, 2018.

[42] Ari Kobren, Barna Saha, and Andrew McCallum. Paper matching with local fairness constraints. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pages 1247–1257, New York, NY, USA, 2019. ACM.

[43] Albert O Hirschman. The paternity of an index. *The American economic review*, 54(5):761–762, 1964.

[44] Andrew V. Goldberg and Tomasz Radzik. A heuristic improvement of the bellman-ford algorithm, 1993.

[45] Boris Cherkassky, Andrew V. Goldberg, and Tomasz Radzik. Shortest paths algorithms: Theory and experimental evaluation. *Mathematical Programming*, 73:129–174, 1993.

[46] Saba Ahmadi, Faez Ahmed, John P. Dickerson, Mark Fuge, and Samir Khuller. An algorithm for multi-attribute diverse matching. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint*

Conference on Artificial Intelligence, IJCAI-20, pages 3–9. International Joint Conferences on Artificial Intelligence Organization, 7 2020. Main track.

[47] 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*, 2018.

[48] Melanie Schmidt, Chris Schwiegelshohn, and Christian Sohler. Fair coresets and streaming algorithms for fair k-means clustering. *arXiv preprint arXiv:1812.10854*, 2018.

[49] Matthäus Kleindessner, Samira Samadi, Pranjal Awasthi, and Jamie Morgenstern. Guarantees for spectral clustering with fairness constraints. *arXiv preprint arXiv:1901.08668*, 2019.

[50] Sara Ahmadian, Alessandro Epasto, Ravi Kumar, and Mohammad Mahdian. Clustering without over-representation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 267–275, 2019.

[51] 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*, volume 108, pages 4195–4205, 26–28 Aug 2020.

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

[53] Moses Charikar, Venkatesan Guruswami, and Anthony Wirth. Clustering with qualitative information. *Journal of Computer and System Sciences*, 71(3):360–383, 2005.

[54] 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*, pages 219–228. ACM, 2015.

[55] Amir Ben-Dor, Ron Shamir, and Zohar Yakhini. Clustering gene expression patterns. *Journal of computational biology*, 6(3-4):281–297, 1999.

[56] Jiong Guo, Falk Hüffner, Christian Komusiewicz, and Yong Zhang. Improved algorithms for bicluster editing. In Manindra Agrawal, Dingzhu Du, Zhenhua Duan, and Angsheng Li, editors, *Theory and Applications of Models of Computation*, pages 445–456, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[57] Vladimir Filkov and Steven Skiena. Integrating microarray data by consensus clustering. *International Journal on Artificial Intelligence Tools*, 13(04):863–880, 2004.

[58] Toshihiro Kamishima, Shotaro Akaho, Hideki Asoh, and Jun Sakuma. Fairness-aware classifier with prejudice remover regularizer. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 35–50. Springer, 2012.

[59] 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*, pages 259–268, 2015.

[60] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*, pages 214–226, 2012.

[61] L Elisa Celis, Damian Straszak, and Nisheeth K Vishnoi. Ranking with fairness constraints. *arXiv preprint arXiv:1704.06840*, 2017.

[62] Clemens Rösner and Melanie Schmidt. Privacy preserving clustering with constraints. *arXiv preprint arXiv:1802.02497*, 2018.

[63] L Elisa Celis, Lingxiao Huang, and Nisheeth K Vishnoi. Multiwinner voting with fairness constraints. *arXiv preprint arXiv:1710.10057*, 2017.

[64] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, and Sergei Vassilvtiskii. Matroids, matchings, and fairness. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2212–2220, 2019.

[65] Gregory J Puleo and Olgica Milenkovic. Correlation clustering and biclustering with locally bounded errors. *IEEE Transactions on Information Theory*, 64(6):4105–4119, 2018.

[66] Saba Ahmadi, Samir Khuller, and Barna Saha. Min-max correlation clustering via multicut. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 13–26. Springer, 2019.

[67] Nir Ailon, Noa Avigdor-Elgrabli, Edo Liberty, and Anke Van Zuylen. Improved approximation algorithms for bipartite correlation clustering. *SIAM Journal on Computing*, 41(5):1110–1121, 2012.

[68] Moses Charikar, Venkatesan Guruswami, and Anthony Wirth. Clustering with qualitative information. In *Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on*, pages 524–533. IEEE, 2003.

[69] Erik D Demaine, Dotan Emanuel, Amos Fiat, and Nicole Immorlica. Correlation clustering in general weighted graphs. *Theoretical Computer Science*, 361(2-3):172–187, 2006.

[70] Alexander Kirillov, Evgeny Levinkov, Bjoern Andres, Bogdan Savchynskyy, and Carsten Rother. Instancecut: From edges to instances with multicut. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 7322–7331. IEEE, 2017.

[71] Sungwoong Kim, Sebastian Nowozin, Pushmeet Kohli, and Chang D. Yoo. Higher-order correlation clustering for image segmentation. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 1530–1538. Curran Associates, Inc., 2011.

[72] Zoya Svitkina and Éva Tardos. Min-max multiway cut. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 207–218. Springer, 2004.

[73] Nikhil Bansal, Uriel Feige, Robert Krauthgamer, Konstantin Makarychev, Viswanath Nagarajan, Seffi Naor, and Roy Schwartz. Min-max graph partitioning and small set expansion. *SIAM Journal on Computing*, 43(2):872–904, 2014.

[74] Eden Chlamtac, Konstantin Makarychev, and Yury Makarychev. How to play unique games using embeddings. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 687–696. IEEE, 2006.

[75] Michael Brückner and Tobias Scheffer. Stackelberg games for adversarial prediction problems. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD 11, page 547555, New York, NY, USA, 2011. Association for Computing Machinery.

[76] Moritz Hardt, Nimrod Megiddo, Christos Papadimitriou, and Mary Wootters. Strategic classification. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, ITCS 16, page 111122, New York, NY, USA, 2016. Association for Computing Machinery.

[77] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[78] Jinshuo Dong, Aaron Roth, Zachary Schutzman, Bo Waggoner, and Zhiwei Steven Wu. Strategic classification from revealed preferences. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, EC 18, page 5570, New York, NY, USA, 2018. Association for Computing Machinery.

[79] Yiling Chen, Yang Liu, and Chara Podimata. Learning strategy-aware linear classifiers. *arXiv preprint arXiv:1911.04004*, 2020.

[80] Lily Hu, Nicole Immorlica, and Jennifer Wortman Vaughan. The disparate effects of strategic manipulation. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, FAT* '19, pages 259–268, New York, NY, USA, 2019. ACM.

[81] Mark Braverman and Sumegha Garg. The role of randomness and noise in strategic classification. In *1st Symposium on Foundations of Responsible Computing (FORC 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

[82] Smitha Milli, John L. Miller, Anca D. Dragan, and Moritz Hardt. The social cost of strategic classification. In *FAT*, 2018.

[83] Jon M. Kleinberg and Manish Raghavan. How do classifiers induce agents to invest effort strategically? In *EC '19*, 2018.

[84] Nika Haghtalab, Nicole Immorlica, Brendan Lucier, and Jack Wang. Maximizing welfare with incentive-aware evaluation mechanisms. In *29th International Joint Conference on Artificial Intelligence*, 2020.

[85] Tal Alon, Magdalen Dobson, Ariel D Procaccia, Inbal Talgam-Cohen, and Jamie Tucker-Foltz. Multiagent evaluation mechanisms. In *AAAI*, pages 1774–1781, 2020.

[86] Yahav Bechavod, Katrina Ligett, Zhiwei Steven Wu, and Juba Ziani. Causal feature discovery through strategic modification. *arXiv preprint arXiv:2002.07024*, 2020.

[87] Yonadav Shavit, Benjamin Edelman, and Brian Axelrod. Causal strategic linear regression. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[88] John Miller, Smitha Milli, and Moritz Hardt. Strategic classification is causal modeling in disguise. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[89] Saba Ahmadi, Hedyeh Beyhaghi, Avrim Blum, and Keziah Naggita. The strategic perceptron. *arXiv preprint arXiv:2008.01710*, 2020.

[90] Vaggos Chatziafratis, Neha Gupta, and Euiwoong Lee. Inapproximability for local correlation clustering and dissimilarity hierarchical clustering. *arXiv preprint arXiv:2010.01459*, 2020.

[91] Matthäus Kleindessner, Pranjal Awasthi, and Jamie Morgenstern. A notion of individual fairness for clustering. *arXiv preprint arXiv:2006.04960*, 2020.