Correlation Clustering in Data Streams

Abstract

Clustering is an fundamental tool for analyzing large data sets and a rich body of work has been devoted to designing data stream algorithm for the relevant optimization problems such as k-center, k-median, and k-means. To be useful, such algorithms need to be both time-efficient and space-efficient. In this paper, we address the problem of *correlation clustering* in the dynamic data stream model: the stream consists of updates to the edge weights of a graph on n nodes and the goal is to find a node-partition such that the end-points of negative-weight edges are typically in different clusters whereas the end-points of positive-weight edges are typically in the same cluster. We present polynomial-time, $O(n \cdot \text{polylog}\, n)$-space algorithms for the natural approximation problems that arise.

We first develop data structures based on linear sketches that allow the "quality" of a given node-partition to be measured. We then use these data structures in combination with convex programming and sampling techniques to solve the relevant approximation problem. The challenges are that the standard LP and SDP formulations are not solvable in $O(n \cdot \text{polylog}\, n)$-space and that the required sampling procedures are often adaptive. Our work presents space-efficient algorithms for the required convex programming and approaches to reduce the adaptivity of the sampling.

1. Introduction

Correlation Clustering. Similar to the contemporaneous Cluster Editing problem (Shamir et al., 2004), the Correlation Clustering problem was first formulated as an optimization problem by Bansal et al. (Bansal et al., 2004). The input is a complete weighted graph G on n nodes, where each pair of nodes uv has weight $w_{uv} \in \mathbb{R}$. A positive-weight edge indicates that u and v should be in the same cluster whereas a if the weight is negative, then u and v should be in different clusters. Given a node-partition $\mathcal{C} = \{C_1, C_2, \ldots\}$, we say edge uv *agrees* with $\mathcal{C}$ if $w_{uv} \geq 0$ and nodes u, v belong to the same cluster or $w_{uv} \leq 0$ and nodes u, v belong to different clusters. The goal is to find the partition $\mathcal{C}$ that maximizes

$$\mathsf{agree}(G, \mathcal{C}) := \sum_{uv \text{ agrees with } \mathcal{C}} |w_{uv}|\,,$$

or, equivalently, that minimizes $\mathsf{disagree}(G, \mathcal{C}) := \sum_{uv} |w_{uv}| - \mathsf{agree}(G, \mathcal{C})$. Solving this problem exactly is known to be NP-hard and a large body of work(Ailon et al., 2008; Bansal et al., 2004; Charikar et al., 2005; Coleman et al., 2008; Giotis & Guruswami, 2006; Swamy, 2004) has been devoted to approximating $\max_{\mathcal{C}} \mathsf{agree}(G, \mathcal{C})$ and $\min_{\mathcal{C}} \mathsf{disagree}(G, \mathcal{C})$. If all weights are ± 1, there is a polynomial time approximation scheme PTAS for max-agree (Bansal et al., 2004; Giotis & Guruswami, 2006) and a 2.5-approximation for min-disagree (Ailon et al., 2008). If the weights are arbitrary, there is a 0.7666-approximation for max-agree (Swamy, 2004) and an $O(\log n)$ approximation for min-disagree (Demaine et al., 2006). If there is an upper bound k on the number of clusters in $\mathcal{C}$ and all weights are ± 1, then a PTAS is known for both problems (Giotis & Guruswami, 2006). However existing linear and semidefinite approaches for min-disagree and max-agree on general graphs require $\omega(n \,\text{polylog}\, n)$ memory even when the input graph is sparse. In particular, the LP for min-disagree has $O(n^2)$ variables and $O(n^3)$ constraints, while the current SDP algorithms for max-agree require $\Omega(n^2)$ space. These typically do not fit in RAM for moderate values of n. Therefore the problem has also been explored from the perspective of avoiding convex programs (Bagon & Galun, 2011; Elsner & Schudy, 2009), little formal analysis is known in space constrained models (see also the tutorial (Bonchi et al., 2014)).

Clustering and Graph Analysis in Data Streams. Given the importance of clustering as a basic tool for analyzing massive data sets, it is unsurprising that a considerable effort has gone into designing clustering algorithms in the relevant computational models. In particular, in the data stream model we are permitted a limited number of passes (ideally just one) over the data while using only limited memory. This model abstracts the challenges in traditional applications of stream processing such as network monitoring and also leads to I/O-efficient external memory algo-

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

rithms. Naturally, for an algorithm to be of use in either context it should also be fast, both in terms of the time to process each stream and returning the final answer.

Classical clustering problems including k-median (Charikar et al., 2003; Guha et al., 2000), k-means (Ailon et al., 2009), and k-center (Charikar et al., 2004; Guha, 2009) have all been studied in the stream model. For a recent survey of the area, see Silva et al. (Silva et al., 2013). However, the first result for correlation clustering was only recently established; Chierichetti et al. (Chierichetti et al., 2014) presented a polynomial-time $(3+\epsilon)$-approximation for min-disagree on ± 1-weighted graphs using $O(\epsilon^{-1}\log^2 n)$ passes. Their basic approach yields both a MapReduce and *semi-streaming* algorithm (Feigenbaum et al., 2005), i.e., a streaming algorithm that uses $\Theta(n \operatorname{polylog} n)$ memory. Using space roughly proportional to the number of nodes can be shown to be necessary for solving many natural graph problems including, it will turn out, correlation clustering. For a recent survey of the semi-streaming algorithms and graph sketching see McGregor (McGregor, 2014).

1.1. Our Techniques and Results

We initiate a formal study of correlation clustering in the data stream model. We start by presenting three basic data structures for the agree and disagree query problems where a partition $\mathcal{C}$ is specified at the end of the stream, and the goal is to return an approximation of $\mathsf{agree}(G, \mathcal{C})$ or $\mathsf{disagree}(G, \mathcal{C})$. These data structures can be constructed in the semi-streaming model and can be queried in $\tilde{O}(n)$ time. They are based on linear sketches and incorporate ideas from recent work on constructing graph sparsifiers via linear sketches (Ahn et al., 2012b; Kapralov et al., 2014). A direct application of these data structures is that it is possible to (with high probability) $(1+\epsilon)$-approximate min-disagree on ± 1-weighted graphs and max-agree on arbitrary weights in the semi-streaming model *if* we are permitted exponential post-processing time. However, our primary use of these data structures will be as primitives in the following *polynomial-time* approximation algorithms:

1. *Maximizing Agreements:* We present a single-pass semi-streaming algorithm that returns a $(1+\epsilon)$-approximation on bounded-weight graphs (see Section 2.2) and a 0.7666-approximation on graphs with arbitrary weights (see Section 3.1).

2. *Minimizing Disagreements:* We present a single-pass $O(\log |E^-|)$-approximation algorithm on graphs with arbitrary weights (see Appendix A). The algorithm uses $\tilde{O}(|E^-| + n)$ space, where E^- is the set of negative weight edges. We prove a matching lower bound on the required space up to factors of $\operatorname{poly}(1/\epsilon, \log n)$ (Lemma 9). At the same time, the multiplicative approximation factor is the best possible factor achievable in polynomial time unless there are better algorithms for the well known multi-cut problem[1]. Finally, in Section 4 we present $O(\log \log n)$-pass algorithms for ± 1-weighted graphs: the approximation factor is $1+\epsilon$ when the number of clusters must be less than some constant, while the factor is 3 in the general case. This last result improves on the result by Chierichetti et al. (Chierichetti et al., 2014)

In particular, our results fully resolve (in terms of matching the best-known non-streaming approximation factors in one-pass using small space) the problem of $\mathsf{max\text{-}agree}(G)$ in the case of ± 1 weights and arbitrary weights and $\mathsf{min\text{-}disagree}(G)$ in the case of arbitrary weights.

Convex Programming in Small Space. While it may appear that we can use linear time SDPs to solve correlation clustering, such an approach would mean that the "alphabet" (the number of clusters in such a reduction) can be large in the general case and a term polynomial in the alphabet size appears in the linear time algorithms for constraint satisfaction problems. Those approaches, as is, would require $\Omega(n^2)$ space. Using quadratic space for post-processing would render the space saved when streaming meaningless!

We introduce several new ideas in solving the LP and SDP for min-disagree and max-agree. For min-disagree we consider a novel formulation as well as switch the role of dual and primal formulations, where the new dual formulation has a natural interpretation for min-disagree. We apply primal-dual algorithms (multiplicative weight update method, see (Arora et al., 2012) for a survey) to the new primal. However the feasible solution of this primal will not shed any light on a feasible (fractional) solution of min-disagree which is the dual. We avoid all this by producing an integral solution for min-disagree and a proof that the primal (fractional) is lower bounded by the integral solution divided by the approximation factor.

We leave open the question of the space complexity of producing a feasible fractional solution for min-disagree and max-agree. For both we provide a direct integral solution and avoid producing a fractional feasible solution. This is in contrast to all previous work on small space algorithms which produce an explicit fractional feasible solution for the target problem (see (Ahn & Guha, 2013) and references therein). We bypass producing such an explicit fractional feasible solution using a general procedure that converts any

[1] In the multi-cut problem where we are given k terminal pairs (s_i, t_i) and the goal is to find the minimum cut that separates each terminal pair. The best known approximation for this problem is $O(\log k)$. Demaine et al. (Demaine et al., 2006) and Charikar et al. (Charikar et al., 2005) provide approximation preserving reductions from multicut to min-disagree.

rounding algorithm to an oracle for the multiplicative weight update method (but applied to the dual for min-disagree). We show that this approach works for the multicut problem as well. For the max-agree problem we do not switch the primal and duals, but use the principles of that general procedure to produce a simple, natural and efficient oracle. Again in this case, we bypass producing a completely feasible fractional solution of max-agree. We expect this general oracle construction idea to be of use for solving other linear and semidefinite programming problems in small space. Due to the space constraints and the necessity to review substantial notation of the multiplicative weight update method (which is mostly orthogonal from the correlation clustering problem) we relegate the discussion of min-disagree to Appendix A. We discuss max-agree (which is shorter) and show the simple oracle in Section 3.

1.2. Preliminary Definitions and Notation

Notation. We consider three types of weighted graphs: (a) *unit weights*, where all $w_{uv} \in \{-1, 1\}$, (b) *bounded weights*, where all $w_{uv} \in [-w_*, -1] \cup [1, w_*]$ for some constant $w_* > 1$, and (c) *arbitrary weights* where $w_{uv} \in \mathbb{R}$. We denote the sets of positive-weight and negative-weight edges by E^+ and E^- respectively and define $G^+ = (V, E^+)$ and $G^- = (V, E^-)$. Let $\Gamma^+(v)$ denote the set of v's neighbors in G^+. Let

$$\mathsf{min\text{-}disagree}(G) = \min_{\mathcal{C}} \mathsf{disagree}(G, \mathcal{C})$$

$$\mathsf{max\text{-}agree}(G) = \max_{\mathcal{C}} \mathsf{agree}(G, \mathcal{C}),$$

and let $\mathsf{min\text{-}disagree}_k(G)$ and $\mathsf{max\text{-}agree}_k(G)$ be the corresponding values when $\mathcal{C}$ ranges over only partitions with at most k clusters. The n^{th} Bell number, $B_n \leq n^n$, is the number of possible partitions of n nodes.

Computational Model. The elements of the stream have the form $(u_i v_i, \Delta_i, s(u_i v_i)) \in \binom{V}{2} \times \mathbb{R} \times \{-1, 1\}$ where $\sum_{i:u_i v_i = uv} \Delta_i \geq 0$ for all uv. The stream then defines an input graph where uv has weight:

$$w_{uv} = \begin{cases} +\sum_{i:u_i v_i = uv} \Delta_i & \text{if } s(uv) = 1 \\ -\sum_{i:u_i v_i = uv} \Delta_i & \text{if } s(uv) = -1 \end{cases}$$

For example, if $s(uv) = 1$ then $uv \in E^+$ and the weight of uv can be incremented and decremented by stream updates but will always satisfy $w_{uv} \geq 0$. Similarly for $s(uv) = -1$. The input stream and all post-processing must be performed in $O(n \operatorname{polylog} n)$ space. Unless indicated otherwise, the algorithm is permitted only a single pass over the input stream.

2. Basic Data Structures and Applications

We introduce three basic data structures that can be constructed with a single-pass over the input stream that defines the weighted graph G. Given a query partition $\mathcal{C}$, these data structures return estimates of $\mathsf{agree}(G, \mathcal{C})$ or $\mathsf{disagree}(G, \mathcal{C})$. Unfortunately, directly using these data structures to solve the correlation clustering optimization problem would require exponential time or $\omega(n \operatorname{polylog} n)$ space. Instead, we will make more careful use of them to design more efficient solutions. In this section, we present a short application of each data structure that illustrates how these data structures can be used as part of solving the optimization problem. We will then develop these applications further in Sections 3 and 4.

2.1. First Data Structure: Bilinear Sketch

Consider first the standard correlation setting with a complete graph with unit weights ($w_{ij} \in \{-1, 1\}$). Given such a graph G and a clustering $\mathcal{C}$ define the matrices M^G and $M^{\mathcal{C}}$ where:

$$m^G_{ij} = \max(0, w_{ij})$$

$$m^{\mathcal{C}}_{ij} = \begin{cases} 0 & \text{if } i \text{ and } j \text{ are separated in } \mathcal{C} \\ 1 & \text{if } i \text{ and } j \text{ are not separated in } \mathcal{C} \end{cases}$$

and observe that the (squared) matrix distance induced by the Frobenius norm gives exactly $\mathsf{disagree}(G, \mathcal{C}) = \|M^G - M^{\mathcal{C}}\|_F^2 = \sum_{ij} (m^G_{ij} - m^{\mathcal{C}}_{ij})^2$. To efficiently estimate $\|M^G - M^{\mathcal{C}}\|_F^2$ when $\mathcal{C}$ is not known apriori, we can repurpose the bilinear sketch approach of Indyk and McGregor (Indyk & McGregor, 2008):

1. Let $\alpha \in \{-1, 1\}^n$ and $\beta \in \{-1, 1\}^n$ be independent random vectors whose entries are 4-wise independent, and in a single pass over the input compute $Y = \sum_{ij \in E^+} \alpha_i \beta_j$.
2. Given query partition $\mathcal{C} = \{C_1, C_2, \ldots\}$, return $X = \left(\sum_\ell \left(\sum_{i \in C_\ell} \alpha_i\right) \left(\sum_{i \in C_\ell} \beta_i\right) - Y\right)^2$.

Analysis. We first show that the time to query a specific clustering is $\tilde{O}(n)$ rather than $\tilde{O}(n^2)$. Observe that the sketch permits the range-efficient update of coordinates corresponding to an $a \times b$ combinatorial rectangle in $\tilde{O}(a+b)$ time. A cluster $C_i \in \mathcal{C}$, on n_i nodes, corresponds to an $n_i \times n_i$ combinatorial rectangle in $M^{\mathcal{C}}$, and hence the total query time is $\tilde{O}(\sum_i n_i) = \tilde{O}(n)$ as claimed. We next argue that repeating the above scheme a small number of times in parallel yields a good estimate of $\mathsf{disagree}(G, \mathcal{C})$. Bounds on the expectation and variance of the estimate follow from (Braverman et al., 2010; Indyk & McGregor, 2008):

Lemma 1. *For each* $\{f_{ij}\}_{i,j \in [n]}$, $\mathbb{E}\left[\left(\sum_{i,j} \alpha_i \beta_j f_{ij}\right)^2\right] = \sum_{i,j} f_{ij}^2$ *and* $\mathbb{V}\left[\left(\sum_{i,j} \alpha_i \beta_j f_{ij}\right)^2\right] \leq 9\left(\sum_{i,j} f_{ij}^2\right)^2$.

Applying the above lemma to $f_{ij} = m^G_{ij} - m^{\mathcal{C}}_{ij}$ establishes that $\mathbb{E}[X] = \mathsf{disagree}(G, \mathcal{C})$ and $\mathbb{V}[X] \leq 9(\mathsf{disagree}(G, \mathcal{C}))^2$. Hence, running $O(\epsilon^{-2} \log \delta^{-1})$ parallel repetitions of the scheme and averaging the results appropriately[2] yields a $(1 \pm \epsilon)$-approximation for $\mathsf{disagree}(G, \mathcal{C})$ with probability at least $1 - \delta$.

Theorem 2. *For unit weights, there exists a $O(\epsilon^{-2} \log \delta^{-1} \log n)$-space algorithm for the* disagree *query problem. Each positive edge is processed in $\tilde{O}(\epsilon^{-2})$ time, while the query time is $\tilde{O}(\epsilon^{-2} n)$.*

The next simple corollary follows by setting $\delta = 1/(nB_n)$ and using the above data structure to evaluate all B_n possible node-partitions.

Corollary 3. *For unit weights, there exists an exponential-time, single-pass stream algorithm, using $\tilde{O}(\epsilon^{-2} n)$ space, that with high probability $(1 + \epsilon)$ approximates* $\mathsf{min\text{-}disagree}(G)$.

In subsequent sections we restrict focus on polynomial-time algorithms, however we note the above simple algorithm is already space-optimal. Furthermore, the restriction to unit weights was essential since any algorithm for general weights requires $\Omega(n^2)$ space. See Appendix D for the proof of this and numerous subsequent theorems.

Theorem 4. *A one-pass stream algorithm that tests whether* $\mathsf{min\text{-}disagree}(G) = 0$*, with probability at least $9/10$, requires $\Omega(n^2)$ bits if weights are arbitrary, and requires $\Omega(n)$ bits even with unit weights.*

Application to Cluster Repair. Consider the Cluster Repair problem (Gramm et al., 2005), in which we are promised $\mathsf{min\text{-}disagree}(G) \leq k$ for some constant k. There is a simple polynomial-time application of the above data structure, as we can narrow down the number of possible clusterings to $\mathrm{poly}(n)$:

1. Construct a spanning forest F of G^+ using the $\tilde{O}(n)$-space dynamic graph algorithm due to Ahn et al. (Ahn et al., 2012a). Let $\mathcal{C}_F$ be the node-partition corresponding to the connected components of F.

2. Let $F_1, F_2, \ldots$ be all the forests formed by deleting at most k edges from F. Let $\mathcal{C}_{F_i}$ be the node-partition corresponding to the connected components of F_i.

Lemma 5. *The optimal partition of G is a refinement of $\mathcal{C}_F$ and a coarsening of some $\mathcal{C}_{F_i}$; there are at most $O\left((n(k+1))^{k+1}\right)$ such partitions.*

[2]Specifically, take the standard approach of partitioning the estimates into $O(\log \delta^{-1})$ groups, each of size $O(\epsilon^{-2})$. With constant probability, the mean of each group is within a $1 \pm \epsilon$ factor; we finally return the median of the resulting group estimates.

Therefore, setting $\delta = O((n(k+1))^{-(k+1)})$ in Theorem 2 yields the following theorem.

Theorem 6. *For unit weight graphs with* $\mathsf{min\text{-}disagree}(G) \leq k$*, there exists a poly-time data-stream algorithm using $\tilde{O}(n + k\epsilon^{-2})$ space that $(1 + \epsilon)$ approximates* $\mathsf{min\text{-}disagree}(G)$ *with high probability.*

2.2. Second Data Structure: Sparsification

The next data structure is based on graph sparsification and works for arbitrarily weighted graphs. A sparsification of graph G is a weighted graph H such that the weight of every cut in H is within a $1 + \epsilon$ factor the weight of the corresponding cut in G. A celebrated result of Benczür and Karger (Benczúr & Karger, 1996) shows that the size of H is at most $\tilde{O}(n\epsilon^{-2})$. A recent result shows that this can be constructed in the streaming model:

Theorem 7 ((Ahn et al., 2012b))**.** *There is a single-pass semi-streaming algorithm that returns a sparsification using space $\tilde{O}(n\epsilon^{-2})$ and time $\tilde{O}(m)$.*

We next show that that a sparsifier yields an estimate of $\mathsf{agree}(G, \mathcal{C})$ or $\mathsf{disagree}(G, \mathcal{C})$ for every partition $\mathcal{C}$.

Lemma 8. *Let H^+ and H^- be sparsifications of G^+ and G^- such that all cuts are preserved within factor $(1 \pm \epsilon/3)$, and let $H = H^+ \cup H^-$. For every clustering $\mathcal{C}$,* $\mathsf{agree}(G, \mathcal{C}) = (1 \pm \epsilon)\mathsf{agree}(H, \mathcal{C}) \pm \epsilon w(E^+)$ *and* $\mathsf{disagree}(G, \mathcal{C}) = (1 \pm \epsilon)\mathsf{disagree}(H, \mathcal{C}) \pm \epsilon w(E^-)$.

Note that $\mathsf{max\text{-}agree}(G) \geq w(E+)$ by considering the trivial all-in-one-cluster partition. Therefore, any near-optimal multiplicative solution for $\mathsf{max\text{-}agree}(H)$ is also a near-optimal multiplicative approximation for $\mathsf{max\text{-}agree}(G)$. We will use this fact in Section 3.1. A weaker result holds for min-disagree that we will use in Appendix A: Given the $\tilde{O}(n)$ positive weights in H^+ and all $|E^-|$ negative weight we can $(1 + \epsilon)$-approximate $\mathsf{disagree}(G, \mathcal{C})$ for every partition $\mathcal{C}$ including the partition that realizes $\mathsf{min\text{-}disagree}(G)$. The following lemma establishes that this approach is essentially optimal.

Lemma 9. *For arbitrary weights, any stream algorithm that determines whether* $\mathsf{min\text{-}disagree}(G) = 0$ *with probability at least $9/10$ requires $\Omega(n + |E^-|)$ bits of space.*

Application to Maximizing Agreements in Unit Weight Graphs. In Section 3.1, we develop a $\mathrm{poly}(n)$-time stream algorithm based on the sparsification construction. That algorithm will returns a 0.766 approximation for max-agree when G has arbitrary weights. However, in the case of unit weights, a RAM-model PTAS for max-agree is known (Bansal et al., 2004; Giotis & Guruswami, 2006). It would be unfortunate if, by approximating the unit-weight graph by a weighted sparsification, we lost the ability to return a $1 \pm \epsilon$ approximation in polynomial time.

We resolve this as follows. We emulate part of an algorithm by Giotis and Guruswami (Giotis & Guruswami, 2006) for max-agree$_k$ using a single pass over the stream[3]. In their algorithm, the nodes are partitioned into $m = O(1/\epsilon)$ groups $V_1, V_2, \ldots, V_m$, each of size $O(\epsilon n)$, and for each V_i we draw a sample of $r = \text{poly}(1/\epsilon, k, \log 1/\delta)$ nodes S_i from $V \setminus V_i$. Using the weights on edges between each S_i and V_i, we generate all possible $(B_r)^m$ partitions. The required sampling can be performed simultaneously with the construction of the sparsifier. Then, at the end of the stream, the possible partitions are generated and we use the graph sparsifier to find the best of these partitions. See Appendix B for further details and a generalization to the case of bounded weights.

Theorem 10. *For bounded weight inputs, there exists a poly-time semi-streaming algorithm that $(1+\epsilon)$-approximates* max-agree(G) *with high probability.*

2.3. Third Data Structure: Node-Based Sketch

In this section we develop a data structure that supports queries to disagree$(G, \mathcal{C})$ for arbitrarily *weighted* graphs when $\mathcal{C}$ is restricted to be a 2-partition (a standard clustering paradigm). For each node i, define the vector, $a^i \in \mathbb{R}^{\binom{n}{2}}$, indexed over $\binom{[n]}{2}$, where the only non-zeros are:

$$a^i_{ij} = \begin{cases} w_{ij}/2 & \text{if } ij \in E^- \\ w_{ij}/2 & \text{if } ij \in E^+, i < j \\ -w_{ij}/2 & \text{if } ij \in E^+, i > j \end{cases}$$

Lemma 11. *For a two-partition $\mathcal{C} = \{C_1, C_2\}$,* disagree$(G, \mathcal{C}) = \| \sum_{\ell \in C_1} a^\ell - \sum_{\ell \in C_2} a^\ell \|_1$.

Hence, we use the ℓ_1-sketching result of Kane et al. (Kane et al., 2010) to compute a random linear sketch of each a^i.

Theorem 12. *For arbitrary weights and query partitions that contain two clusters, to solve the* disagree *query problem, there exists an $O(\epsilon^{-2} n \log \delta^{-1} \log n)$-space algorithm. The query time is $O(\epsilon^{-2} n \log \delta^{-1} \log n)$.*

It is natural to ask whether this approach can be extended to queries $\mathcal{C}$ where $|\mathcal{C}| > 2$. The following lemma proves that this is unfortunately not the case.

Lemma 13. *When $|\mathcal{C}| = 3$, a data structure that returns a multiplicative estimate of* disagree$(G, \mathcal{C})$ *with probability at least $9/10$, requires $\Omega(n^2)$ space.*

Application to Two-Center Min-Disagreement Clustering with Bounded Weights. We apply the above node-based sketch in conjunction with another algorithm by Giotis-Guruswami, this time for min-disagree$_2$. Their algorithm samples $r = \text{poly}(1/\epsilon) \cdot \log n$ nodes S and using the weights of the edges incident on S generates 2^{m-1} possible partitions. Appendix B describes the generalization of that algorithm to the bounded weights case, and other implementation details. The sampling of S and the incident edges can be performed using one-pass and $O(nr \log n)$ space. We then find the best of these possible partitions in post-processing using the above node-based sketches.

Theorem 14. *For bounded weight inputs, there exists a poly-time semi-streaming algorithm that $(1+\epsilon)$-approximates* min-disagree$_2(G)$ *with high probability.*

3. Max Agree in General Weighted Graphs

In this section, we present a $0.7666(1-\epsilon)$-approximation algorithm for max-agree. In the RAM model, there are algorithms for max-agreebased on semi-definite programming (Charikar et al., 2005; Swamy, 2004). Of relevance, Swamy developed a 0.7666-approximation algorithm based on this SDP:

$$\begin{aligned} \max \quad & \tfrac{1}{2}\Big[\textstyle\sum_{(i,j)\in E^+} w_{ij}(\|x_i\|^2 + \|x_j\|^2 - \|x_i - x_j\|^2) \\ & \quad + \textstyle\sum_{(i,j)\in E^-} |w_{ij}| \|x_i - x_j\|^2\Big] \\ \text{s.t.} \quad & \|x_i\|^2 = 1 \text{ for all } i \in V \\ & x_i \cdot x_j \geq 0 \text{ for all } i, j \in V \end{aligned} \tag{SDP$_{\text{MA}}$}$$

If two vertices, i and j, are in the same cluster, their corresponding vectors x_i and x_j will coincide, so $\|x_i - x_j\| = 0$; on the other hand, if they are in different clusters, their vectors should be orthogonal.

To be space efficient, we apply Swamy's algorithm to the semi-streaming model along with the SDP feasibility algorithm by Steurer (Steurer, 2010). The outline is as follows.

1. Sparsify the graph, preserving agreements within factor $1 \pm \delta$, using $m' = \tilde{O}(n\delta^{-2})$ edges (Section 2.2).
2. Solve SDP$_{\text{MA}}$ approximately, which requires a separation oracle (Section 3.1) that finds a set of violated constraints. This involves guessing the optimal value of the SDP, α. Note that it is trivial to find a $\frac{1}{2}$-approximation of the maximum agreement using a random partition of the graph. Letting W_s be the total weight of edges in H, the sparsified graph, we perform binary search over α approximated to powers of $(1+\delta)W_s$. This increases the running time by a $O(\log \delta^{-1})$ factor.

   With this oracle, we do not guarantee $x_i \cdot x_j \geq 0$ in the fractional solution: we only guarantee $x_i \cdot x_j \geq -\delta$. Ensuring $x_i \cdot x_j \geq 0$ appears to be difficult (or require a substantially different oracle).
3. Even though the standard rounding algorithm (Swamy,

[3] Setting $k = O(1/\epsilon)$ is sufficient to yield a $(1+\epsilon)$-approximation for the case when the number of clusters is unrestricted (Bansal et al., 2004).

2004) requires $x_i \cdot x_j \geq 0$, we show in Section 3.1 how to round the fractional solution with $x_i \cdot x_j \geq -\delta$.

Theorem 15. *There is a one-pass $0.7666(1-\epsilon)$-approximation algorithm for the maximizing agreements problem in the semi-streaming model that runs in $\tilde{O}(m + n\epsilon^{-8})$ time.*

3.1. The SDP Feasibility Algorithm

Steurer's SDP feasibility algorithm (Steurer, 2010) is based on the matrix multiplicative-weights method by Arora and Kale (Arora & Kale, 2007). We use the notation of both papers for the canonical formulation of a decision SDP.

Definition 1. *For matrices $\mathbf{A}, \mathbf{B}$, let $\mathbf{A} \circ \mathbf{B}$ denote $\sum_{i,j} \mathbf{A}_{ij}\mathbf{B}_{ij}$. Let $\mathbf{A} \succeq 0$ denote that $\mathbf{A}$ is (positive) semidefinite, and let $\mathbf{A} \succeq \mathbf{B}$ denote that $\mathbf{A} - \mathbf{B}$ is (positive) semidefinite. A semidefinite decision (denoted by ?) problem is in* **canonical form** *if it can be written as:*

$$?\mathbf{C} \circ \mathbf{X} \geq \alpha\,, \qquad \mathbf{A}_j \circ \mathbf{X} \leq b_j \ \text{ for all } j\,, \qquad \mathbf{X} \succeq 0\,,$$

where $\mathbf{C} \circ \mathbf{X} \geq \alpha$ is a special constraint that corresponds to the (maximization) objective value of the SDP. We denote the set of the feasible solutions by $\mathcal{X}$.

In SDP_{MA} the vectors x_i define the semidefinite matrix $\mathbf{X}$: $\mathbf{X}_{ij} = x_i \cdot x_j$. For every decision SDP in the canonical form, the feasibilty algorithm consists of multiple iterations of two-party game. In each iteration, an oracle of the feasibility algorithm is given a candidate $\mathbf{X}$ and does exactly one of the following:

- *Either:* Declares the SDP feasible and returns a feasible (either fractional or integral) solution;
- *Or:* Returns a hyperplane $(\mathbf{A}, b)$ that separates $\mathbf{X}$ from a feasible region.

If the oracle returns a feasible solution, the process stops; otherwise, it updates $\mathbf{X}$ (we use Steurer's update procedure (Steurer, 2010)) and continues to the next iteration. If there exists a feasible solution, the sequence of candidates (the $\mathbf{X}$s) keeps moving toward the feasible region, and eventually the oracle cannot find a suitable separating hyperplane and therefore returns an intended solution to the SDP. Formally, we have the following definitions and theorem.

Definition 2. *Define $d_i = \sum_{(i,j)} |w_{ij}|$ and $\sum_i d_i = 2W$. Let $\mathbf{D}$ be the diagonal matrix with $\mathbf{D}_{ii} = d_i/2W$.*

Definition 3. *Let $\mathcal{X}$ be the set of feasible solutions of some SDP. Suppose that given a candidate $\mathbf{X}$ the separation oracle returns a hyperplane $(\mathbf{A}, b)$ (if it returns a hyperplane). Given nonnegative δ and ρ, the separation oracle is defined to be:*

1. *δ-separating iff $\mathbf{A} \circ \mathbf{X} \leq b - \delta$ and $\mathbf{A} \circ \mathbf{X}' \geq b$ for all $\mathbf{X}' \in \mathcal{X}$, and*
2. *ρ-bounded iff $-\rho\mathbf{D} \preceq \mathbf{A} - b\mathbf{D} \preceq \rho\mathbf{D}$,*

where $\mathbf{A} \preceq \mathbf{B}$ is the shorthand for $\mathbf{A} \circ \mathbf{Y} \leq \mathbf{B} \circ \mathbf{Y}$ for every positive semidefinite $\mathbf{Y}$.

Theorem 16. *If $\mathcal{X}$ is non-empty, the oracle outputs a feasible (fractional or integral) solution within $O((\rho^2 \log n)/\delta^2)$ iterations using Steurer's weight update procedure (Steurer, 2010).*

The update procedure (Steurer, 2010) maintains (and defines) the candidate vector $\mathbf{X}$ implicitly. In particular it uses matrices of dimension $n \times d$, in which every entry is a (scaled) Gaussian random variable. The algorithm also uses a precision parameter (degree of the polynomial approximation to represent matrix exponentials) r. Assuming that T_M is the time for a multiplication between a returned $\mathbf{A}$ and some vector, the update process computes the tth $\mathbf{X}$ in time $O(t \cdot r \cdot d \cdot T_M)$, a quadratic dependence on t in total. We will ensure that any returned $\mathbf{A}$ has at most m' nonzero entries, and therefore $T_M = O(m')$. The space required by the algorithm is the space required to represent a linear combination of the matrices $\mathbf{A}$ which are returned in the different iterations. In the following we show that $\rho = O(1/\delta)$ and using Theorem 16 the total number of iterations is $\tilde{O}(\delta^{-4})$. For our purposes, $d = O((\log n)/\delta^2)$, $r = O(\log(1/\delta))$, and $T_M = O(m')$, giving us a $\tilde{O}(n\delta^{-10})$ time and $\tilde{O}(n\delta^{-2})$ space algorithm.

Space-Time Tradeoffs and Open Questions. Unlike the general $\mathbf{X}$ used in Steurer's approach, in our oracle the $\mathbf{X}$ is used in a very specific way by the Oracle. In particular, we only need to test whether (i) $\mathbf{X}_{ij} < -\delta$, (ii) $\mathbf{X}_{ii} \geq 1 + \delta$, (iii) $\mathbf{X}_{ii} \leq 1 - \delta$, and (iv) whether a linear function of $\mathbf{X}_{ij}$ is at most $(1 - 4\delta)\alpha$. Therefore we can use L_2 sketches for the linear combination of the returned matrices $\mathbf{A}$, which defines the implicit representation of the n vectors x_i that define $\mathbf{X}$. The running time of this type of oracle can be made $\tilde{O}(m'\delta^{-2})$ for every iteration instead of the previous dependence on the number of previous iterations (Steurer, 2010). The oracle first estimates all $x_i \cdot x_i$, and all $x_i \cdot x_j$ for (i,j) in the sparsified graph, and then performs the computation as denoted by Algorithm 1. Therefore the running time would reduce to $O(\delta^{-4})$ times $\tilde{O}(m'\delta^{-2})$ which is $\tilde{O}(n\delta^{-8})$.

However for two different iterations t, t' even though the $\mathbf{X}(t), \mathbf{X}(t')$ are related, it is easiest to sketch them independently which implies that the number of sketches will be $\tilde{O}(\delta^{-4})$ (the iterations) times $\tilde{O}(m'\delta^{-2})$ (size of sketch). The space required is therefore also $\tilde{O}(n\delta^{-8})$. It appears that the dependence across the iterations is mild and it may be possible to use $O(m'\delta^{-2})$ space. This leaves open the

question of determining the exact space-versus-running-time tradeoff. We now discuss the Oracle and the rounding algorithms.

The Oracle The Oracle is proviced in Algorithm 1. This is a slight alteration of the general procedure in Algorithm 3 in Appendix A.1 here we are using all the violated constraints instead of the single constraint in Algorithm 3. However the overall structure is exactly the same.

Algorithm 1 Oracle for SDP_{MA}.

1: For the separating hyperplane, we only describe non-zero entries in $\mathbf{A}$. Recall that we have a candidate $\mathbf{X}$ where $\mathbf{X}_{ij} = x_i \cdot x_j$.
2: Let $S_1 = \{i : \|x_i\|^2 \geq 1 + \delta\}$ and $\Delta_1 = \sum_{i \in S_1} d_i$.
3: Let $S_2 = \{i : \|x_i\|^2 \leq 1 - \delta\}$ and $\Delta_2 = \sum_{i \in S_2} d_i$.
4: Let $S_3 = \{(i,j) : x_i \cdot x_j < -\delta\}$ and $\Delta_3 = \sum_{(i,j) \in S_3} |w_{ij}|$.
5: **if** $\Delta_1 \geq \delta\alpha$ **then**
6: Let $\mathbf{A}_{ii} = -d_i/\Delta_1$ for $i \in S_1$ and $b = -1$. Return $(\mathbf{A}, b)$.
7: **else if** $\Delta_2 \geq \delta\alpha$ **then**
8: Let $\mathbf{A}_{ii} = d_i/\Delta_2$ for $i \in S_2$ and $b = 1$. Return $(\mathbf{A}, b)$.
9: **else if** $\Delta_3 \geq \delta\alpha$ **then**
10: Let $\mathbf{A}_{ij} = w_{ij}/\Delta_3$ for $(i,j) \in S_3$ and $b = 0$. Return $(\mathbf{A}, b)$.
11: **else**
12: Ignore all nodes in S_1 and S_2 and all edges in S_3. Let $\mathbf{C}'$ be the matrix that corresponds to the objective function of the modified graph G'.
13: **if** $\mathbf{C}' \circ \mathbf{X} < (1 - 4\delta)\alpha$ **then**
14: Let $\mathbf{A} = \mathbf{C}'/\alpha$ and $b = 1 - 3\delta$. Return $(\mathbf{A}, b)$.
15: **else**
16: Round $\mathbf{X}$, as described in Section 3.1, and return the rounded solution.

Lemma 17. *Algorithm 1 is δ-separating.*

Proof. For line 3.1, $\mathbf{A} \circ \mathbf{X} \leq \sum_{i \in S_1} -d_i(1+\delta)/\Delta_1 = -1 - \delta$, since $\|x_i\|^2 \geq 1 + \delta$ for all $i \in S_1$. On the other hand, for a feasible $\mathbf{X}'$, $\|x'_i\|^2 = 1$ for all i. Hence $\mathbf{A} \circ \mathbf{X}' = \sum_{i \in S_1} -d_i/\Delta_1 = -1$. This proves that the oracle is δ-separating when it returns from line 3.1. For lines 3.1 and 3.1, the proof is almost identical.

For line 3.1, we do not use the violated constraints; instead we use $\mathbf{C}'$ to construct $\mathbf{A}$, and show that $\mathbf{C}' \circ \mathbf{X}' \geq (1 - 3\delta)\alpha$. We start from the fact that $\mathbf{C} \circ \mathbf{X}' \geq \alpha$, since $\mathbf{X}'$ is feasible for SDP_{MA}. By removing all nodes in S_1, we remove all edges incident on the removed nodes. The total weight of removed edges is bounded by Δ_1, which is this case is less than $\delta\alpha$. Similarly, we lose at most $\delta\alpha$ for each of S_2 and S_3. Hence, the difference between $\mathbf{C}' \circ \mathbf{X}'$ and $\mathbf{C} \circ \mathbf{X}'$ is bounded by $3\delta\alpha$, and so $\mathbf{C}' \circ \mathbf{X}' \geq (1 - 3\delta)\alpha$ which implies $\mathbf{A} \circ \mathbf{X}' \geq 1 - 3\delta$. Therefore we have δ separation because, $\mathbf{A} \circ \mathbf{X} = \mathbf{C}' \circ \mathbf{X}/\alpha < 1 - 4\delta$. □

Lemma 18. *Algorithm 1 is ρ-bounded for $\rho = O(1/\delta)$.*

Proof. Since $|b| \leq 1$ in each case, to prove $-\rho\mathbf{D} \preceq \mathbf{A} - b\mathbf{D} \preceq \rho\mathbf{D}$, it suffices to show that for every positive semidefinite $\mathbf{Y}$, $|\mathbf{A} \circ \mathbf{Y}| = \rho\mathbf{D} \circ \mathbf{Y}$. For line 3.1, the proof is straightforward. To start, $\mathbf{A}$ is a diagonal matrix where $|\mathbf{A}_{ii}| = d_i/\Delta_1 \leq d_i/(\delta\alpha)$. On the other hand, $\mathbf{D}_{ii} = d_i/2W$, while $\alpha \geq W/2$, so we have $|\mathbf{A}_{ii}| = O(1/\delta)\mathbf{D}_{ii}$ which proves that $|\mathbf{A} \circ \mathbf{Y}| = O(1/\delta)\mathbf{D} \circ \mathbf{Y}$. The proof is identical for line 3.1.

For lines 3.1 and 3.1, we use the fact that $y_i \cdot y_j \leq \|y_i\|^2 + \|y_j\|^2$ for every pair of vectors y_i and y_j. Therefore for $\mathbf{Y}_{ij} = y_i \cdot y_j$, we have at line 3.1,

$$\begin{aligned}
|\mathbf{A} \circ \mathbf{Y}| &= \sum_{(i,j) \in S_3} \frac{|w_{ij}|}{\Delta_3} \mathbf{Y}_{ij} \\
&\leq \sum_{(i,j) \in S_3} \frac{|w_{ij}|}{\Delta_3} (\|y_i\|^2 + \|y_j\|^2) \\
&= \frac{1}{\Delta_3} \sum_i \|y_i\|^2 \sum_{j:(i,j) \in S_3} |w_{ij}| \\
&\leq \frac{1}{\Delta_3} \sum_i d_i \|y_i\|^2 \\
&= \frac{1}{\Delta_3} \sum_i 2W\mathbf{D}_{ii}\mathbf{Y}_{ii} = \frac{2W}{\Delta_3} \mathbf{D} \circ \mathbf{Y}\,,
\end{aligned}$$

which implies $|\mathbf{A} \circ \mathbf{Y}| \leq O(1/\delta)\mathbf{D} \circ \mathbf{Y}$ given $\alpha \geq W/2$ and $\Delta_3 \geq \delta\alpha$. For line 3.1,

$$\begin{aligned}
&\mathbf{A} \circ \mathbf{Y} = \frac{1}{\alpha} \mathbf{C}' \circ \mathbf{Y} \\
&= \frac{1}{2\alpha} \left(\sum_{(i,j) \in E^+|_{G'}} 2w_{ij}\mathbf{Y}_{ij} + \sum_{(i,i) \in E^-|_{G'}} |w_{ij}|(\mathbf{Y}_{ii} + \mathbf{Y}_{jj} - 2\mathbf{Y}_{ij}) \right) \\
&\leq \frac{1}{2\alpha} \sum_{(i,j) \in G'} 2|w_{ij}|(\mathbf{Y}_{ii} + \mathbf{Y}_{jj}) \leq \frac{1}{\alpha} \sum_i d_i \mathbf{Y}_{ii} = \frac{2W}{\alpha} \mathbf{D} \circ \mathbf{Y}
\end{aligned}$$

which implies that $\mathbf{A} \circ \mathbf{Y} = O(1)\mathbf{D} \circ \mathbf{Y}$. Summarizing, Algorithm 1 is $O(1/\delta)$-bounded. □

Rounding the Fractional Solution. The following Lemma proves the rounding algorithm which makes the overall framework possible.

Lemma 19. *If Algorithm 1 returns a clustering solution, it has at least $0.7666(1 - O(\delta))\alpha$ agreements.*

Proof. We show that the rounding algorithm returns a clustering with at least $0.7666(1 - O(\delta))\mathbf{C}' \circ \mathbf{X}$ agreements. Combined with the fact that $\mathbf{C}' \circ \mathbf{X} > (1 - 4\delta)\alpha$ (line 3.1), we obtain the desired result.

Since we deal with $\mathbf{C}'$ instead of $\mathbf{C}$, we can ignore all nodes and edges in S_1, S_2, and S_3. We first rescale the vectors in $\mathbf{X}$ to be unit vectors. Since all vectors that are not ignored (not in S_1 nor S_2) have length between $1-O(\delta)$ and $1+O(\delta)$ (since we take the square root), this only changes the objective value by $O(\delta w_{ij})$ for each edge. Hence the total decrease is bounded by $O(\delta W) = O(\delta\alpha)$.

We then (1) first change the objective value of edges (i,j) with $-\delta < x_i \cdot x_j < 0$ by ignoring them, and only then (2) consider fixing the violated constraints $x_i \cdot x_j < 0$ to produce a feasible or integral solution.

Step (1) decreases the objective value by at most $\delta|w_{ij}|$ for each negative edge. Again, the objective value decreases by at most $O(\delta\alpha)$. For step (2) we use Swamy's rounding algorithm (Swamy, 2004), which obtains a 0.7666 approximation factor. The constraint $x_i \cdot x_j \geq 0$ required by Swamy's algorithm is not satisfied for some edges. However, the rounding algorithm is based on random hyperplanes and the probability that x_i and x_j are split by a hyperplane only increases as $x_i \cdot x_j$ decreases. For positive edges, we already accounted for this in step (1) when the value of the edge was made 0. For negative edges, the probability that i and j land in different clusters only increases by having negative $x_i \cdot x_j$, but again, the contribution to the objective is still 0. Therefore, we obtain a clustering that has at least $0.7666(1-O(\delta))\mathbf{C}' \circ \mathbf{X}$ agreements. □

4. Multipass Algorithms

In this section, we present $O(\log\log n)$-pass algorithms for min-disagree on unit weight graphs where either the number of clusters is fixed or unrestricted. The fixed number of clusters is dicussed in Appendix C.

4.1. Minimizing Disagreements for Unit Weights

Consider the following 3-approximation algorithm for min-disagree on unit-weight graphs due to Ailon et al. (Ailon et al., 2008):

1: Index the nodes V as $v_1, \ldots, v_n$ in a random order. Let $U \leftarrow V$ be the set of "uncovered" nodes.
2: **for** $i = 1$ to n **do**
3: **if** $v_i \in U$ **then**
4: Let v_i be "chosen" and define $C_i \leftarrow \{v_i\} \cup \{v_j \in U : v_i v_j \in E^+\}$ and let $U \leftarrow U \setminus C_i$.
5: **else**
6: $C_i \leftarrow \emptyset$.
7: **Return** the collection of non-empty sets C_i.

It may appear that emulating the above algorithm in the data stream model requires $\Omega(n)$ passes, since determining if v_i should be chosen depends on whether v_j is chosen for all $j < i$. However, we will show that $O(\log\log n)$-passes suffice. This improves upon a result by Chierichetti et al. (Chierichetti et al., 2014), who developed a modification of the algorithm that used $O(\epsilon^{-1}\log^2 n)$ streaming passes and returned a $3+\epsilon$-approximation rather than a 3-approximation. Our improvement is based on the following lemma:

Lemma 20. *Define $F^+_{t,t'} = \{v_i v_j \in E^+, i,j \in U_t, t < i,j \leq t'\}$ where U_t is the set of uncovered nodes after iteration t of the above algorithm. Then, with high probability, $|F^+_{t,t'}| \leq 5 \cdot \ln n \cdot t'^2/t$.*

Proof. Fix an arbitrary node $v \in V_{t,t'} = \{v_{t+1}, \ldots, v_{t'}\}$. For $i \in [t]$, define $d_i = |U_{i-1} \cap \Gamma^+(v)\}|$ to be the number of uncovered neighbors of v immediately before the ith iteration of the algorithm. By the principle of deferred decisions, the probability that $v_i \in \Gamma(v)$ is an uncovered neighbor of v at iteration i is at least d_i/t'.

So, either there is some $i \in [t]$ for which d_i drops below $10 \cdot \ln n \cdot t'/t$ or the probability that no neighbor of v is chosen as a center is at most $(1-(10 \cdot \ln n \cdot t'/t)/t')^t \leq 1/n^{10}$. Hence, with high probability each uncovered node in $V_{t,t'}$ has at most $10 \cdot \ln n \cdot t'/t$ uncovered neighbors in $V_{t,t'}$. Remembering that nodes are counted twice, the bound follows by summing over all t' nodes. □

Semi-Streaming Algorithm. Our semi-streaming algorithm proceeds as follows. For $j \geq 1$ let $t_j = (2n)^{1-1/2^j}$ and during the $(2j-1)$-th pass we collect all edges in $F^+_{t_{j-1},t_j}$ and during the $(2j)$-th pass we can determine U_{t_j}. Note that at the end of the $(2j)$-th pass we are able to simulated the first t_j iterations of Ailon et al.'s algorithm. Since $t_j \geq n$ for $j = 1 + \log\log n$, our algorithm terminates after $O(\log\log n)$ passes.

Theorem 21. *On a unit-weighted graph, there exists a $O(\log\log n)$-pass semi-streaming algorithm that returns with high probability a 3-approximation to* min-disagree.

Proof. For the odd numbered passes, by Lemma 20, the space is at most

$$5 \cdot \ln n \cdot t_j^2/t_{j-1} = 5 \cdot \ln n \cdot 2n = O(n \log n)\,,$$

with high probability. The additional space used in the even numbered passes is trivially bounded by $O(n\log n)$. The approximation factor follows from the analysis of Ailon et al. (Ailon et al., 2008). □

References

Ablayev, Farid M. Lower bounds for one-way probabilistic communication complexity and their application to space complexity. *Theor. Comput. Sci.*, 157(2):139–159, 1996. doi: 10.1016/0304-3975(95)00157-3. URL http://dx.doi.org/10.1016/0304-3975(95)00157-3.

Ahn, Kook Jin and Guha, Sudipto. Access to data and number of iterations: Dual primal algorithms for maximum matching under resource constraints. *CORR, arXiv 1307.4359*, 2013.

Ahn, Kook Jin, Guha, Sudipto, and McGregor, Andrew. Analyzing graph structure via linear measurements. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 459–467, 2012a. URL http://portal.acm.org/citation.cfm?id=2095156&CFID=63838676&CFTOKEN=79617016.

Ahn, Kook Jin, Guha, Sudipto, and McGregor, Andrew. Graph sketches: sparsification, spanners, and subgraphs. In *ACM Principles of Database Systems (PODS)*, pp. 5–14, 2012b. doi: 10.1145/2213556.2213560. URL http://doi.acm.org/10.1145/2213556.2213560.

Ailon, Nir, Charikar, Moses, and Newman, Alantha. Aggregating inconsistent information: Ranking and clustering. *J. ACM*, 55(5), 2008. doi: 10.1145/1411509.1411513. URL http://doi.acm.org/10.1145/1411509.1411513.

Ailon, Nir, Jaiswal, Ragesh, and Monteleoni, Claire. Streaming k-means approximation. In *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada.*, pp. 10–18, 2009. URL http://books.nips.cc/papers/files/nips22/NIPS2009_1085.pdf.

Arora, Sanjeev and Kale, Satyen. A combinatorial, primal-dual approach to semidefinite programs. In *ACM Symposium on Theory of Computing (STOC)*, pp. 227–236, 2007.

Arora, Sanjeev, Hazan, Elad, and Kale, Satyen. The multiplicative weights update method: a meta algorithm and applications. *Theory of Computing*, 8(6):121–164, 2012. doi: 10.4086/toc.2012.v008a006. URL http://www.theoryofcomputing.org/articles/v008a006.

Bagon, S. and Galun, M. Large scale correlation clustering optimization. *arXiv:1112.2903v1*, 2011.

Bansal, Nikhil, Blum, Avrim, and Chawla, Shuchi. Correlation clustering. *Machine Learning*, 56(1-3):89–113, 2004. doi: 10.1023/B:MACH.0000033116.57574.95. URL http://dx.doi.org/10.1023/B:MACH.0000033116.57574.95.

Benczúr, András A. and Karger, David R. Approximating $s-t$ minimum cuts in $\tilde{O}(n^2)$ time. In *ACM Symposium on Theory of Computing (STOC)*, pp. 47–55, 1996.

Bonchi, Francesco, Garcia-Soriano, David, and Liberty, Edo. Correlation clustering: From theory to practice. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pp. 1972–1972, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2956-9. doi: 10.1145/2623330.2630808. URL http://doi.acm.org/10.1145/2623330.2630808.

Braverman, Vladimir, Chung, Kai-Min, Liu, Zhenming, Mitzenmacher, Michael, and Ostrovsky, Rafail. AMS without 4-wise independence on product domains. In *International Symposium on Theoretical Aspects of Computer Science (STACS)*, pp. 119–130, 2010. doi: 10.4230/LIPIcs.STACS.2010.2449. URL http://dx.doi.org/10.4230/LIPIcs.STACS.2010.2449.

Charikar, Moses, O'Callaghan, Liadan, and Panigrahy, Rina. Better streaming algorithms for clustering problems. In *ACM Symposium on Theory of Computing (STOC)*, pp. 30–39, 2003. doi: 10.1145/780542.780548. URL http://doi.acm.org/10.1145/780542.780548.

Charikar, Moses, Chekuri, Chandra, Feder, Tomás, and Motwani, Rajeev. Incremental clustering and dynamic information retrieval. *SIAM J. Comput.*, 33(6):1417–1440, 2004. doi: 10.1137/S0097539702418498. URL http://dx.doi.org/10.1137/S0097539702418498.

Charikar, Moses, Guruswami, Venkatesan, and Wirth, Anthony. Clustering with qualitative information. *J. Comput. Syst. Sci.*, 71(3):360–383, 2005. doi: 10.1016/j.jcss.2004.10.012. URL http://dx.doi.org/10.1016/j.jcss.2004.10.012.

Chierichetti, Flavio, Dalvi, Nilesh N., and Kumar, Ravi. Correlation clustering in mapreduce. In *SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 641–650, 2014. doi: 10.1145/2623330.2623743. URL http://doi.acm.org/10.1145/2623330.2623743.

Coleman, Tom, Saunderson, James, and Wirth, Anthony. A local-search 2-approximation for 2-correlation-clustering. In *Algorithms - ESA 2008, 16th Annual European Symposium, Karlsruhe, Germany, September 15-17, 2008. Proceedings*, pp. 308–319, 2008. doi: 10.1007/978-3-540-87744-8_26. URL http://dx.doi.org/10.1007/978-3-540-87744-8_26.

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

Elsner, Micha and Schudy, Warren. Bounding and comparing methods for correlation clustering beyond ilp. In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, ILP '09, pp. 19–27, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-35-0. URL http://dl.acm.org/citation.cfm?id=1611638.1611641.

Feigenbaum, Joan, Kannan, Sampath, McGregor, Andrew, Suri, Siddharth, and Zhang, Jian. On graph problems in a semi-streaming model. *Theoretical Computer Science*, 348(2-3):207–216, 2005. URL http://dx.doi.org/10.1016/j.tcs.2005.09.013.

Garg, Naveen, Vazirani, Vijay V., and Yannakakis, Mihalis. Approximate max-flow min-(multi)cut theorems and their applications. In *ACM Symposium on Theory of Computing (STOC)*, pp. 698–707, 1993.

Giotis, Ioannis and Guruswami, Venkatesan. Correlation clustering with a fixed number of clusters. *Theory of Computing*, 2(1):249–266, 2006. doi: 10.4086/toc.2006.v002a013. URL http://dx.doi.org/10.4086/toc.2006.v002a013.

Gramm, Jens, Guo, Jiong, Hüffner, Falk, and Niedermeier, Rolf. Graph-modeled data clustering: Exact algorithms for clique generation. *Theory Comput. Syst.*, 38(4):373–392, 2005. doi: 10.1007/s00224-004-1178-y. URL http://dx.doi.org/10.1007/s00224-004-1178-y.

Guha, Sudipto. Tight results for clustering and summarizing data streams. In *International Conference on Database Theory (ICDT)*, pp. 268–275, 2009. doi: 10.1145/1514894.1514926. URL http://doi.acm.org/10.1145/1514894.1514926.

Guha, Sudipto, Mishra, Nina, Motwani, Rajeev, and O'Callaghan, Liadan. Clustering data streams. In *IEEE Foundations of Computer Science (FOCS)*, pp. 359–366, 2000. doi: 10.1109/SFCS.2000.892124. URL http://doi.ieeecomputersociety.org/10.1109/SFCS.2000.892124.

Indyk, Piotr and McGregor, Andrew. Declaring independence via the sketching of sketches. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 737–745, 2008. URL http://dl.acm.org/citation.cfm?id=1347082.1347163.

Kalyanasundaram, Bala and Schnitger, Georg. The probabilistic communication complexity of set intersection. *SIAM J. Discrete Math.*, 5(4):545–557, 1992. doi: 10.1137/0405044. URL http://dx.doi.org/10.1137/0405044.

Kane, Daniel M., Nelson, Jelani, and Woodruff, David P. On the exact space complexity of sketching and streaming small norms. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1161–1178, 2010. doi: 10.1137/1.9781611973075.93. URL http://dx.doi.org/10.1137/1.9781611973075.93.

Kapralov, Michael, Lee, Yin Tat, Musco, Cameron, Musco, Christopher, and Sidford, Aaron. Single pass spectral sparsification in dynamic streams. *CoRR*, abs/1407.1289, 2014. URL http://arxiv.org/abs/1407.1289.

McGregor, Andrew. Graph stream algorithms: a survey. *SIGMOD Record*, 43(1):9–20, 2014. doi: 10.1145/2627692.2627694. URL http://doi.acm.org/10.1145/2627692.2627694.

Shamir, Ron, Sharan, Roded, and Tsur, Dekel. Cluster graph modification problems. *Discrete Applied Mathematics*, 144(1):173–182, 2004.

Silva, Jonathan A., Faria, Elaine R., Barros, Rodrigo C., Hruschka, Eduardo R., Carvalho, André C. P. L. F. de, and Gama, João. Data stream clustering: A survey. *ACM Comput. Surv.*, 46(1):13:1–13:31, July 2013. ISSN 0360-0300. doi: 10.1145/2522968.2522981. URL http://doi.acm.org/10.1145/2522968.2522981.

Steurer, David. Fast sdp algorithms for constraint satisfaction problems. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 684–697, 2010.

Swamy, Chaitanya. Correlation clustering: maximizing agreements via semidefinite programming. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 526–527, 2004.