Evaluating Stability in Massive Social Networks: Efficient Streaming Algorithms for Structural Balance

Vikrant Ashvinkumar
Department of Computer Science, Rutgers University, New Brunswick, NJ, USA

Sepehr Assadi
Department of Computer Science, Rutgers University, New Brunswick, NJ, USA
Cheriton School of Computer Science, University of Waterloo, Canada

Chengyuan Deng
Department of Computer Science, Rutgers University, New Brunswick, NJ, USA

Jie Gao
Department of Computer Science, Rutgers University, New Brunswick, NJ, USA

Chen Wang
Department of Computer Science, Rutgers University, New Brunswick, NJ, USA

Abstract

Structural balance theory studies stability in networks. Given a n-vertex complete graph $G = (V, E)$ whose edges are labeled positive or negative, the graph is considered *balanced* if every triangle either consists of three positive edges (three mutual "friends"), or one positive edge and two negative edges (two "friends" with a common "enemy"). From a computational perspective, structural balance turns out to be a special case of correlation clustering with the number of clusters at most two. The two main algorithmic problems of interest are: (i) detecting whether a given graph is balanced, or (ii) finding a partition that approximates the *frustration index*, i.e., the minimum number of edge flips that turn the graph balanced.

We study these problems in the streaming model where edges are given one by one and focus on *memory efficiency*. We provide randomized single-pass algorithms for: (i) determining whether an input graph is balanced with $O(\log n)$ memory, and (ii) finding a partition that induces a $(1+\varepsilon)$-approximation to the frustration index with $O(n \cdot \text{polylog}(n))$ memory. We further provide several new lower bounds, complementing different aspects of our algorithms such as the need for randomization or approximation.

To obtain our main results, we develop a method using pseudorandom generators (PRGs) to sample edges between independently-chosen *vertices* in graph streaming. Furthermore, our algorithm that approximates the frustration index improves the running time of the state-of-the-art correlation clustering with two clusters (Giotis-Guruswami algorithm [SODA 2006]) from $n^{O(1/\varepsilon^2)}$ to $O(n^2 \log^3 n/\varepsilon^2 + n \log n \cdot (1/\varepsilon)^{O(1/\varepsilon^4)})$ time for $(1+\varepsilon)$-approximation. These results may be of independent interest.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms; Theory of computation → Pseudorandomness and derandomization

Keywords and phrases Streaming algorithms, structural balance, pseudo-randomness generator

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.58

Category RANDOM

Related Version *Full Version*: https://arxiv.org/abs/2306.00668

Funding *Sepehr Assadi*: Assadi is supported in part by a Alfred P. Sloan Research Fellowship, and Assadi and Wang are supported in part by a NSF CAREER Grant CCF-2047061, a gift from Google Research, and a Rutgers Research Council Fulcrum Award.

Jie Gao: Ashvinkumar, Deng and Gao have been supported by NSF Grant CCF-2118953, CRCNS-2207440, CCF-2208663, and CNS-2137245.

Acknowledgements We thank Karthik C.S. for some preliminary discussions.

1 Introduction

Structural balance theory [21, 30, 35, 36] arises in the study of social relationships with positive and negative relations. Positive links describe friendship/agreement and negative links describe antagonism/disagreement. The theory dates back to work by Heider [35]. It describes the stability of relations among three individuals – only two kinds of triangles are stable: the ones where all three ties are positive, indicating mutual friendship; and the ones with two negative ties and one positive tie, describing the folklore that "the enemy of your enemy is your friend." A network that is far from being balanced (i.e., with many unstable triangles) accumulates stress which may lead to major re-arrangements of edges.

The structural balance theory appears in many application areas such as international relations [7, 43], biological networks [29, 38], portfolio analysis in financial networks [33], Ising model [14] in statistical physics, and online social media and opinion formation, dynamics, and evolution [6, 48–50] which bears itself on, for example, Facebook [47] and Twitter [40, 51].

One fundamental question is to understand whether a network is close to being balanced or not. In a *complete signed* graph where each pair of vertices is labeled positive or negative – the focus of our study – , the *Cartwright-Harary Theorem* [21, 32] states that every balanced network must have the nodes partitioned into (at most) two "camps", inside each of which the edges are all positive and between them the edges are negative. For graphs that are not balanced, a natural measure to characterize its distance from total balance is the *frustration index*, defined as the minimum number of edges whose negation of signs results in balance [1, 8, 34, 53]. These questions can be distilled into the following algorithmic problems (see Problem 1 and Problem 2 for the formal definitions):

- **Structural Balance Testing**: Given a complete signed graph G, decide whether or not it is balanced, namely, does not contain any imbalanced triangle.
- **Frustration-minimizing Partition**: Given a complete signed graph G, find a partition of vertices into two camps such that the minimum number of sign flips on the edges is required for the resulting graph to be balanced.

The problems we consider are closely related to (min-disagreement) *correlation clustering* where the goal is to partition the graph into clusters, so as to minimize the total number of negative edges inside clusters and positive edges across the clusters [5, 13, 24, 25, 28, 31], except that structural balance enforces the number of clusters to be two. Structural balance testing is straightforward to solve in $O(n^2)$ time on n-vertex graphs: place an arbitrary vertex and all its positive neighbors on one side of the bi-partition L, and its negative neighbors on the other side R, and then verify. On the other hand, the classical work by Giotis and Guruswami [31] on correlation clustering with two clusters implies the NP-hardness of the frustration-minimizing partition. It further provides a PTAS with running time $n^{O(1/\varepsilon^2)} + n \cdot (1/\varepsilon)^{O(1/\varepsilon^4)}$ for $(1+\varepsilon)$-approximation of this problem. To our knowledge, this is the state of the art for structural balance testing and frustration-minimizing partition.

Prior work primarily focused on *running time* of algorithms and assume *unrestricted access* to the entire graph. In many modern applications of large-scale networks, however, there are many other considerations to take into account. For instance, the algorithms may only have **limited access** to and a **memory** much smaller than the input. One of the most popular models capturing the above scenario is the **graph streaming model**. In this model, the edges arrive one after another and the algorithm needs to process this stream "on-the-fly"

with limited memory. The algorithm is allowed to make one or multiple passes over the stream, and the memory is usually substianlly smaller than the (worst-case) input size, i.e. $\Theta(n^2)$. We focus on the single-pass setting and ask the following motivating question: *How well can we solve structural balance testing and frustration-minimizing partition problems in the single-pass graph streaming setting?*

The golden spots for streaming algorithms are (i) the polylog(n) memory regime, which is polynomial to the memory that is necessary to represent a single edge, and (ii) the $\widetilde{O}(n) := O(n \cdot \text{polylog}(n))$ memory regime – often referred to as the *semi-streaming* model. We study structural balance in the graph streaming model in these parameter regimes.

1.1 Our Contributions

Structural balance testing. One way of testing structural balance of G is to check whether the 2-lift of G associated with its edge signing is connected. Computing the 2-lift can be done on-the-fly, and testing connectivity in the streaming model can be done with $O(n \log n)$ space. This is already a quadratic improvement over the trivial algorithm with $O(n^2)$ space, and it is also known that graph connectivity requires $\Omega(n \log n)$ space in the streaming model. Is this also the limit for structural balance testing? Our first result is a simple algebraic algorithm, which stems from testing for complete bipartiteness of negative edges, that uses *exponentially improved* space complexity. In the full version, we also provide a companion algorithm that, while more complicated to describe, is combinatorial and introduces a technique that partially derandomizes a vertex sketch we use where hash functions with limited-independence alone do not suffice.

▶ **Result 1** (Informal statement of Theorem 2). *There is a single-pass randomized algorithm that given a complete signed graph G in a graph stream, tests whether G is balanced with probability at least $\frac{99}{100}$ using $O(\log n)$ space.*

Result 1 shows that structural balance testing can be done with high space efficiency – note that $\Theta(\log n)$ space is necessary to simply write down a single edge. As we elaborate later, our algorithms in Result 1 are *sketching-based* algorithms that are randomized and crucially use the fact that each edge of the graph appears precisely once in the stream. We further complement our algorithms with two new lower bounds (Propositions 10 and 11) that show the necessity of both conditions for obtaining any $o(n)$-space algorithm for this problem. Additionally, a standard $\Omega(n \log n)$ space lower bound for input graphs that are not necessarily complete via a reduction from bipartiteness testing can be shown (see the full version); the generalized problem and the $o(n)$ space regime are thus incompatible.

Frustration-minimizing partition. Outputting the solution to frustration-minimizing partition already requires $\Omega(n)$ space, thus, as is standard, we focus on semi-streaming algorithms for this problem. One can use *cut sparsifiers* and a variant of an argument by [3] (for correlation clustering) to obtain an algorithm with $\widetilde{O}(n/\varepsilon^2)$ space that can return the "frustration value" of any bi-partition of vertices to within a $(1 \pm \varepsilon)$ factor. By enumerating over all bi-partitions, then, we can find the frustration-minimizing one up to a $(1 + \varepsilon)$-approximation. The problem with this approach is that the resulting algorithm takes exponential time (to enumerate all bi-partitions), which is quite prohibitive, especially in large graphs[1].

[1] This scenario is not uncommon in the streaming model. For instance, the Tournament Feedback Arc Set problem admits an "easy" exponential-time $(1 + \varepsilon)$-approximation semi-streaming algorithm [22] that was improved very recently by [16] to a PTAS albeit with $O(\log n)$ passes over the stream; see [15] for another example.

To bypass this challenge, we present a "streamified" version of the correlation clustering algorithm with two clusters of [31] (henceforth, the Giotis-Guruswami algorithm); namely, an approach that allows us to collect just enough information from the stream to *weakly* simulate (meaning not entirely faithfully) the Giotis-Guruswami algorithm in polynomial time at the end of the stream.

▶ **Result 2** (Informal statement of Theorem 7). *There is a single-pass randomized algorithm that given a complete signed graph in a graph stream, with high probability*[a] *finds a partition of vertices with frustration value at most* $(1+\varepsilon)$ *factor of the frustration index using* $\widetilde{O}(n/\varepsilon^2)$ *space and polynomial (in* n *but not* ε*) time.*

[a] Here, and throughout, with high probability means with probability at least $1 - 1/n$.

Result 2 gives an efficient semi-streaming algorithm for $(1+\varepsilon)$-approximation of the frustration-minimizing partition problem, which is NP-hard to solve exactly. We further show (Proposition 13) that even if we allow the streaming algorithm to use exponential (or more) time, solving this problem exactly requires $\Omega(n^2)$ space (the same as storing the entire input). This fully rules out any non-trivial streaming algorithms for solving this problem exactly.

There is still one missing piece in obtaining "truly efficient" semi-streaming algorithms for our problem. As stated earlier, the Giotis-Guruswami algorithm that Result 2 builds on has running time (roughly) $n^{O(1/\varepsilon^2)}$ even ignoring any streaming aspects[2], which makes this algorithm quite impractical. Our final contribution remedies this state of affairs. By building on our weak simulation of the Giotis-Guruswami algorithm in Result 2, we design an improved offline (non-streaming) algorithm for frustration-minimizing partition with nearly-linear running time for any fixed $\varepsilon > 0$ (note that the input is of size $\Theta(n^2)$ in this problem over a complete signed graph).

▶ **Result 3** (Informal statement of Theorem 9). *There is a randomized (classical) algorithm that given a complete signed graph, with high probability, finds a partition of vertices with frustration value at most* $(1+\varepsilon)$ *factor of the frustration index in* $\widetilde{O}(n^2/\varepsilon^2 + n\cdot(1/\varepsilon)^{O(1/\varepsilon^4)})$ *time.*

Result 3 presents the first improvement after two decades over the Giotis-Guruswami algorithm for frustration-minimizing partition and correlation clustering with two fixed clusters *outside of graph streaming models*, which can be of independent interest. Moreover, the algorithm in Result 3 can also be used in Result 2, improving the running time to nearly-linear for any constant $\varepsilon > 0$ in the semi-streaming model.

Our algorithmic results combined with our new lower bounds (in Section 5) collectively complete the picture of streaming and efficient algorithms for structural balance testing and frustration-minimizing partition problems.

1.2 Our Techniques

Structural balance testing. The lower bounds in Propositions 10 and 11 show that any $o(n)$-space algorithm for testing structural balance has to be randomized and, more importantly, uses the fact that it sees the sign between every pair of vertices *exactly once*. This motivates

[2] A simple modification of the Giotis-Guruswami algorithm and a slightly more careful analysis actually reduces the running time of their algorithm to roughly $O(n^{100} + n \cdot (1/\varepsilon)^{O(1/\varepsilon^4)})$ (see the analysis in Section 4). While this is faster than the original $n^{O(1/\varepsilon^2)}$ bound, it is still quite far from being practical.

us to consider *sketching-based* algorithms that are based on collecting aggregate statistics of large subsets of edges in the graph when seeing them in the stream (without having to store explicit information). We proceed by proving the following structural result (stated informally): (i) In any balanced graph, for any set S of *odd* size, there is an *even* number of pairs of vertices in S with a negative sign (easy direction); (ii) Conversely, in any unbalanced graph, a constant fraction of odd-size sets S (of certain cardinality) have an *odd* number of negatively-signed pairs inside (hard direction).

This result naturally suggests the following algorithm (stated with some oversimplification): sample an odd-size set S of vertices uniformly at random at the beginning of the stream and count the number of negative edges in the stream with both endpoints in S. In the balanced case, this number will be even, while in the unbalanced case, it has a non-trivial chance of being odd.

There is a serious challenge in making this strategy work: determining S and therefrom which edges should be counted requires $\Omega(n)$ space to *store the random bits*. We surmount this by observing that the function for the parity of "$-$" edges in a sampled set S can be viewed as a *degree-2 polynomial* with one variable per vertex. As such, we turn from sampling vertices to using PRGs for low-degree polynomials based on sums of small bias generators so that $O(\log(n))$ truly random bits and $O(\log(n))$ extra space suffice to generate pseudorandom bits whose 0-set on the polynomial is in proportion with that of truly random bits. In the full version, we also give an alternative combinatorial algorithm (in contrast with the abovestated algebraic algorithm) which is interesting in its own right, especially from a techniques standpoint.

Frustration-minimizing partition. Our algorithm in Result 2 is obtained via "streamifying" the Giotis-Guruswami algorithm [31]. Their algorithm considers the high versus low frustration index cases separately where the high frustration index scenario means at least a constant fraction of all pairs needs to flip sign. Among these, the first part has a simple solution in [31] which already lends itself to a semi-streaming algorithm immediately. Our main technical ingredient in Result 2 lies in addressing the low frustration index case. For simplicity of exposition, in the following, we assume ε is a constant.

The Giotis-Guruswami algorithm in the low frustration index case is as follows. Sample $O(\log n)$ vertices S and enumerate all $n^{O(1)}$ bi-partitions of S. Then, for every bi-partition (S_L, S_R), perform a *merging* operation followed by a *switching* operation: In merging, every vertex $v \in V \setminus S$ is assigned in parallel to the side of (S_L, S_R) that creates less frustration for v, i.e., the number of negative edges of v to this side plus its positive edges to the opposing side is minimized. Let (L, R) be the resulting bi-partition of vertices. In switching, each vertex $v \in V$ in parallel is re-assigned to the side of (L, R) with the least frustration. The analysis of [31] is as follows: for the bi-partition (S_L, S_R) of S that is consistent with the optimal solution, (i) after merging, most vertices are in the "correct" side of (L, R) already, namely, they are consistent with the optimal bi-partition, and (ii) after switching, the vertices already consistent with the optimal solution do not change side, and the rest of vertices are moved to the side that only induce a small additive cost. The key reasoning behind both steps is that in the low frustration-index case, most vertices have a "clear" preference toward which side of the optimal bi-partition they should reside.

To simulate the Giotis-Guruswami algorithm via a semi-streaming algorithm, we can sample the set S at the beginning of the stream and store all $\widetilde{O}(n)$ edges incident on it during the stream. The merging phase can now be easily implemented as it relies only on the edges connected to S. The switching phase, on the other hand, requires checking *every* edge in

the graph and thus cannot be faithfully implemented in $o(n^2)$ space. We instead develop a way to *approximately* implement this step, by sampling $O(\log n)$ edges per vertex and using them to estimate the frustration of each vertex in the switching phase. This allows us to still classify the majority of vertices the same as that of the Giotis-Guruswami algorithm except for the ones with no clear preference between bi-partitions of the optimal solution. An extra argument here ensures that this step does not increase the cost too much and this new solution is still a $(1+\varepsilon)$-approximation, leading to our semi-streaming algorithm.

The ideas above also allow us to significantly improve the *running time* of the Giotis-Guruswami algorithm even outside of graph streaming models. The most time-consuming step of that algorithm is the need to enumerate all bi-partitions of the sampled set S of size $O(\log n)$. We preface the sampling step of the set S by sampling a smaller set T of size only $O(\log \log n)$, enumerate only over $(\log n)^{O(1)}$ bi-partitions of T, and use those to find an *approximate* optimal bi-partition of the set S. We then follow a similar strategy as above to simulate the Giotis-Guruswami algorithm, by showing that even though the bi-partition of the set S is no longer truly optimal, the extra error occurred by the approximation, does *not* propagate too much in the merging and switching step – in other words, these operations are "robust" enough to handle even an approximately optimal bi-partition of S, not a truly optimal one.

Lower Bounds. Most of our lower bounds are based on reductions from known problems in communication complexity such as Index or Equality. The exception is our lower bound in Proposition 11 which shows that, perhaps counter-intuitively, when a stream contains copies of an edge more than once, solving structural balance testing becomes impossible in $o(n)$ space. We show this lower bound by presenting a new *3-party* communication problem which is a mixture of the Index and Set Intersection problems. Roughly speaking, in this new problem, we also provide the information about the hidden index of the Index problem to *all* players, but in a way that to find this index, they will need to solve an instance of the Set Intersection problem. We then borrow an idea from [9, 10] that allows us to argue that a low-communication protocol cannot even change the distribution of the hidden index by much. We then combine this with standard information-theoretic arguments to show that the Index problem the players need to solve remains hard as the distribution of its hidden index has not been altered too much.

1.3 Related Work

By slightly relaxing structural balance to allow triangles with three negative ties as in [30], the frustration-minimizing partitions becomes fully equivalent to the correlation clustering problem (without any constraint on the number of clusters). More recently, motivated by similar considerations as in our paper, there has been a flurry of results on this problem in modern models of computation such as sublinear-time, streaming, or massively parallel algorithms [3, 11, 17, 18, 26, 27]. In particular, for the single-pass streaming setting, the very recent state-of-the-art result by [23] obtains a $(3+\varepsilon)$-approximation correlation clustering in $O(n/\varepsilon)$ space and polynomial time. Our result shows that we can obtain a better approximation when the number of clusters is two. Furthermore, [18] independently observes a $(1+\varepsilon)$-approximation algorithm for correlation clustering in exponential time using cut sparsifiers, which is similar to our observation in Lemma 8.

In a general graph (not necessarily complete), structural balance means that all cycles must contain an even number of negative edges. The problem of minimizing frustration index, phrased in the literature as the Balanced Subgraph problem [37], is MaxSNP-hard [45, 52]

and even NP-hard to approximate within any constant factor assuming Khot's Unique Games Conjecture [39]. On the positive side, this problem admits a polynomial time $O(\sqrt{\log n})$-approximation [2] and an $O(\log k)$-approximation [12] when the frustration index is k. Readers can refer to [37] for further details about applications in practice. None of these algorithms consider space constraints or the streaming setting.

2 Preliminaries

Notation. Throughout, we use $G = (V, E = E^+ \cup E^-)$ to denote a complete signed graph with $|V| = n$ vertices, where E^+ is the set of "+" edges and E^- is the set of "−" edges. We further use $G^+ = (V, E^+)$ and $G^- = (V, E^-)$ to denote the graph restricted to the "+" and "−" edges. Note that E^+ and E^- are always disjoint and $|E^+ \cup E^-| = \binom{n}{2}$. For any set $S \subseteq V$ of vertices, $G[S]$ denotes the induced subgraph of G on S.

For two sets of vertices $S, T \subset V$, we use $E(S,T)$ to denote the set of edges (both "+" and "−") with one endpoint in S and the other in T. Analogously, we use $E(v,S)$ to denote the set of edges (both "+" and "−") with one endpoint v and the other in S. Furthermore, $E^+(S,T)$ (resp. $E^-(S,T)$) refers to the set of "+" (resp. "−") edges with one endpoint in S and the other in T, and $E^+(v,S)$ (resp. $E^-(v,S)$) refers to the set of "+" (resp. "−") edges with one endpoint v and the other in S. In particular, $N^+(S) = E^+(S, V \setminus S)$ and similarly, $N^+(v) = E^+(v,V)$ ($N^-(S)$ and $N^-(v)$ defined in the same manner).

When the context involves more than one graph (say G and H), we add a subscript to make the notation clear as to which graph is being referred to (for example, $E_G(S,T)$ and $E_H(S,T)$).

2.1 Problem Definition

In this paper, we consider (strong) structural balance. A complete signed graph $G = (V, E^+ \cup E^-)$ is said to exhibit structural balance property (in short, is balanced) if every triangle has an even number of "−" edges. An equivalent characterization of structural balance shown in [21] is that there exists a bi-partition of the vertices into S and T, $S \cap T = \emptyset$ and $S \cup T = V$, such that every edge with both endpoints in S (or T) is labelled "+", and every edge connecting S to T is labelled "−". Inspired by this, the notion of frustration index [1, 34, 53] captures how far a graph is from structural balance.

▶ **Definition 1** (Frustration Index). *Let $G = (V, E^+ \cup E^-)$ be a complete signed graph, and let (L,R) be a bi-partition of V. Then the* frustration *of G with respect to L, R is*

$$\mathsf{frust}(G, L, R) = \left|E^+(L,R)\right| + \left|E^-(L)\right| + \left|E^-(R)\right|.$$

When the context is clear, we use the notation $\mathsf{frust}(G, L)$ *instead of* $\mathsf{frust}(G, L, R)$*. The* frustration index $\mathsf{frust}(G)$ *is the minimum of* $\mathsf{frust}(G, L, R)$ *over all possible bi-partitions (L,R) of V.*

Frustration can be equivalently defined using "disagreement" notation:

$$\mathsf{frust}(G, L, R) = \frac{1}{2} \sum_{v \in V} \mathsf{Dis}(v, L, R).$$

We use $\mathsf{Dis}(v, S, T)$ to count the disagreement of edges incident to v with respect to S, T. For example, when $v \in S$, we have $\mathsf{Dis}(v, S, T) = |E^-(v,S)| + |E^+(v,T)|$. When $T = V \setminus S$, we may write $\mathsf{Dis}(v, S)$ instead of $\mathsf{Dis}(v, S, V \setminus S)$.

Now we are ready to formally define our problems.

▶ **Problem 1** (Structural Balance Testing). *Given a complete signed graph $G = (V, E^+ \cup E^-)$, decide if there is a partition (L, R) of V such that*

- *If $(u, v) \in E^+$, then u and v are both contained in L or both contained in R;*
- *If $(u, v) \in E^-$, then either u is in L and v is in R, or vice versa.*

In other words, the answer is "YES" if and only if the graph is balanced.

▶ **Problem 2** (Frustration-minimizing Partition). *Given a complete signed graph $G = (V, E^+ \cup E^-)$, find a partition (L^*, R^*) of V such that*

$$(L^*, R^*) = \arg\min_{\substack{(L,R):\\ L\cup R=V\\ L\cap R=\{\}}} \mathsf{frust}(G,\ L, R).$$

In other words, the partition (L^*, R^*) minimizes the frustration.

3 Structural Balance Testing in Logarithmic Space

We formally present our main algorithm for structural balance testing (Problem 1) – a randomized algorithm that uses only $O(\log n)$ bits and returns whether the graph is balanced with high (constant) probability.

▶ **Theorem 2** (Formalization of Result 1). *There is a randomized single-pass algorithm* BalanceTester *that solves Problem 1 with $O(\log n)$ bits of space and the following guarantees:*

- *If input graph G is balanced,* BalanceTester *always outputs* Balanced*;*
- *If input graph G is not balanced,* BalanceTester *outputs* NotBalanced *with probability at least $\frac{99}{100}$.*

A few remarks on Theorem 2 are in order. First, it is not hard to design a deterministic algorithm that uses $\tilde{O}(n)$ space to test whether the graph is balanced – for completeness, such an algorithm can be found in the full version of this paper. However, by our lower bound in Proposition 10, any single-pass streaming algorithm for structural balance testing with $o(n)$ memory has to be randomized. Furthermore, we remark that the $O(\log n)$ memory is asymptotically optimal since it is the number of bits that is necessary to store even a single edge. Finally, since the error is one-sided, we can boost the success probability to $1 - \frac{1}{n}$ by running the algorithm $O(\log n)$ times, which results in $O(\log^2 n)$ overall space complexity.

We now proceed to the design and analysis of our algorithm. In what follows, we assume our graph contains at least $n \geq 3$ vertices, as the structural balance is always satisfied otherwise.

3.1 A Sample-and-Test Lemma for "−" Edges

The idea behind our algorithm starts as follows. If we focus on "−" edges, our task can be framed as testing if the graph $G^- = (V, E^-)$ is a *complete bipartite graph*. To this end, observe that if we sample a set S of an odd number of vertices, and count the number of "−" edges in $G[S]$, the parity will always be even when G^- is complete bipartite. With a slightly more involved analysis, we can show that if G^- is not complete bipartite, then by sampling S uniformly at random, the number of "−" edges in $G[S]$ is odd with constant probability. As such, we can use the parity of such a counter as a signal of the structural balance of the graph.

We now formalize the above intuition. In particular, in the following lemma, we design the S-sampler and state its properties.

▶ **Lemma 3** (S-sampler Lemma). *For a given complete signed graph $G = (V, E)$, let $S \subseteq V$ be a subset of vertices sampled with the following S-sampler:*

1. *Pick a fixed vertex v_n arbitrarily.*
2. *Sample each vertex in $v \in V \setminus \{v_n\}$ independently with probability $\frac{1}{2}$ to get S'.*
3. *If $|S'|$ is odd, let $S = S'$; otherwise, let $S = S' \cup \{v_n\}$.*

Then, the following statements are true:

1. *If G is balanced, the induced subgraph $G[S]$ always has an even number of "$-$" edges;*
2. *If G is not balanced, the induced subgraph $G[S]$ has an odd number of "$-$" edges with probability at least $\frac{1}{4}$, over the randomness of sampling S.*

The main idea to prove Lemma 3 is by observing that the parity of "$-$" edges counted by our S-sampler is in fact captured by a *degree-2* polynomial over $\mathbb{F}_2$. The observation can be formalized as the following lemma.

▶ **Lemma 4.** *Let $X_i \in \{0, 1\}$ be the indicator random variable for whether vertex v_i is sampled by S-sampler. Define the following polynomial with $(n-1)$ variables over $\mathbb{F}_2$:*

$$P^*(X) = \sum_{\substack{i,j<n \\ (v_i,v_j)\in E^-}} X_i X_j + \sum_{\substack{i<n \\ (v_i,v_n)\in E^-}} X_i \left(1 + \sum_{j=1}^{n-1} X_j\right). \tag{1}$$

Then, the polynomial $P^*(X) = |E^-(G[S])|$ over $\mathbb{F}_2$, which is the parity of number of negative edges induced by the sample set S.

Proof. Note that Eq (1) is essentially obtained by substituting $X_n = 1 + \sum_{j=1}^{n-1} X_j$, and counting a "$-$" edge (v_i, v_j) if both v_i and v_j are sampled. If the number of sampled vertices other than v_n is odd, we have $\sum_{j=1}^{n-1} X_j = 1$, which implies $X_n = 0$ over $\mathbb{F}_2$; otherwise, if the number of sample vertices other than v_n is even, we have $\sum_{j=1}^{n-1} X_j = 0$, which implies $X_n = 1$ over $\mathbb{F}_2$. Therefore, the polynomial P^* exactly captures the sampling process of the S-sampler. Finally, if the total number of "$-$" edges is odd, the polynomial P^* evaluates to 1, and vice versa. ◀

As we will see shortly, Lemma 4 is also crucial to run our *streaming* algorithm for the S-sampler by storing limited number of random bits. To show that the S-sampler gives a useful signal for whether the graph is balanced, we show that the polynomial P^* in Eq (1) gives a useful signal. We first show that P^* is identically 0 if and only if G is balanced.

▶ **Lemma 5.** *The polynomial P^* in Eq (1) is identically 0 if and only if G is balanced.*

Proof. Suppose first that G is balanced. Then either G^- is empty, or it is a complete bipartite graph. Let S be the sample from S-sampler. In the first case, the number of "$-$" edges in $G[S]$ is always 0. In the second case, let L and R be the bi-partition of G^-, and let $A = L \cap S$ and $B = R \cap S$. The number of "$-$" edges in $G[S]$ is $|A| \cdot |B|$, which is an even number since one of $|A|$ and $|B|$ is even (recall that $|S|$ is odd). By Lemma 4, P^* is thus identically 0 when G is balanced.

Suppose, on the other hand, that G is not balanced. Then there is a triangle (v_i, v_j, v_k) with an odd number of "$-$" edges. If $i, j, k \neq n$, setting $X_i = X_j = X_k = 1$ and $X_\ell = 0$ for $\ell \neq i, j, k$ yields $P^*(X) = 1$. If on the other hand, say, $k = n$, then setting $X_i = X_j = 1$ and $X_\ell = 0$ for $\ell \neq i, j$ yields $P^*(X) = 1$. Hence P^* is not identically 0 when G is not balanced. ◀

Next, we give a lemma on the *fraction* of assignments that are non-zero on the polynomial P^*, which gives us a lower bound for the success probability to capture an odd number of "$-$" edges when G is imbalanced. We note that the lemma is standard in the coding theory community (e.g. [46]), but we state a proof here for completeness.

▶ **Lemma 6.** *Let $P(x_1, x_2, \ldots, x_n)$ be a polynomial over $\mathbb{F}_2$ with degree at most 2. If P is not identically 0, then there is a set X, with cardinality at least 2^{n-2}, such that $x \in X$ implies $P(x) = 1$.*

Proof. We may assume that P is multilinear since $\bar{P}(x) = P(x)$ for all $x \in \{0,1\}^n$, where $\bar{P}$ is the linearization of P (i.e. all monomials of the form x_i^2 are replaced with x_i).

If P is of degree 1 or less, the claim holds by using the Schwartz-Zippel lemma.Suppose then that P is of degree 2. Up to a reordering of variables, $x_{n-1}x_n$ is a term of P with coefficient 1. Let a be a fixed assignment of x_k to $\{0,1\}$ for all $k \in [n-2]$. Then $P(a, x_{n-1}, x_n)$ is a multilinear degree 2 polynomial $Q(x_{n-1}, x_n)$ with the term $x_{n-1}x_n$ having coefficient 1. Checking over all 8 possible polynomials that Q may take, we see that each of them has at least one assignment a' of x_{n-1}, x_n to $\{0,1\}$ such that $Q(a') = 1$. This completes the proof since there are 2^{n-2} possible assignments a. ◀

Finalizing the proof of Lemma 3. By Lemma 4 and Lemma 5, if G is balanced, the polynomial P^* is identically 0, which means the S-sampler always finds an even number of "$-$" edges. Otherwise, if G is not balanced, P^* is not 0. Therefore, by Lemma 6, the S-sampler finds an odd number of "$-$" edges with probability at least $\frac{1}{4}$. ◀

By Lemma 3, to test the structural balance of a complete signed graph, it suffices to implement the S-sampler and count the number of negative edges. However, while the counter takes $O(\log n)$ space, it is not immediately clear how the S-sampler can be implemented with a small space in the streaming setting. In particular, since we need to sample each vertex *independently*, the trivial solution requires $\Omega(n)$ memory to store the random bits (for each vertex). We address this issue in the next step.

3.2 Simulating the S-sampler in Streaming with $O(\log n)$ Space

We tackle this issue by using the fact that our S-sampler is a degree-2 polynomial (Lemma 4). We may consequently use a PRG that fools degree-2 polynomials (see [20, 41, 44] for constructions). More concretely, there exists a PRG that takes $O(\log n)$ truly random bits, and generates n pseudorandom bits with $O(\log n)$ extra space per bit so that a degree-2 polynomial cannot distinguish a truly random input from a pseudorandom input to within a small constant error.

Using this, we design an $O(\log n)$-memory algorithm as follows.

A streaming algorithm to test structural balance.

- Run 100 independent copies of the following S-sampler simulation:
  1. Maintain $O(\log n)$ truly random bits and a counter.
  2. For each arriving "$-$" edge $(u, v) \in E^-$:
     a. Generate the pseudorandom bits for vertices u and v (name the bits X_u and X_v) with a PRG for degree 2 polynomials with $\varepsilon = \frac{1}{20}$ (see [20, 41, 44]).
     b. If both $X_u = 1$ and $X_v = 1$, increase the counter by 1.
  3. By the end of the stream, if the parity of the counter is odd, return 1; otherwise, return 0.
- If any of the copies of S-sampler simulation returns 1, report Not Balanced; otherwise, report Balanced.

Analysis of the space complexity. The truly random bits and the counter take $O(\log n)$ bits of space. Furthermore, for each "$-$" edge, the PRG generates the pseudorandom bits for X_u and X_v with $O(\log n)$ bits of extra space since $\varepsilon = \frac{1}{20}$. The space for this purpose can be re-used across different edges. Each copy of the S-sampler simulation therefore takes $O(\log n)$ space to implement. Finally, since we run 100 independent copies in parallel, the final algorithm take $O(\log n)$ space.

Analysis of correctness. By Lemma 3, if the graph is balanced, the S-sampler simulation always returns 0. On the other hand, if the graph is imbalanced and we sample with truly random bits U_{n-1}, it holds from Lemma 3 that $\Pr(P^*(U_{n-1}) = 1) \geq \frac{1}{4}$ since P^* is a valid implementation of the S-sampler. By the guarantees of the PRG $g : \{0,1\}^{O(\log n) \to (n-1)}$, we have

$$|\Pr(P^*(g(U_\ell)) = 1) - \Pr(P^*(U_{n-1}) = 1)| \leq \frac{1}{20},$$

where U_ℓ are $O(\log n)$ truly random bits, the seed for the PRG. We hence have $\Pr(P^*(g(U_l)) = 1) \geq \frac{1}{4} - \frac{1}{20} = \frac{1}{5}$. As such, the probability for no copy to return 1 is at most $(\frac{4}{5})^{100} < \frac{1}{100}$, as desired.

4 Semi-streaming Frustration-minimizing Partition in Poly-Time

In this section, we consider Problem 2 of finding, in the semi-streaming model, a partition of a complete signed graph that is as close to structurally balanced as possible. More specifically, we want to divide the set of vertices into two parts and minimize the number of "$-$" edges contained in either part and "$+$" edges connecting the two parts.

Giotis and Guruswami [31] showed that Problem 2 is NP-hard. In the streaming setting, we show that any algorithm that gives the exact optimal frustration-minimizing partition (or computes the optimal frustration index) requires $\Omega(n^2)$ memory (see Proposition 13). In view of this, we settle for finding a partition that approximates one which is as close to structurally balanced as possible. This section shows how to adapt the correlation clustering algorithm with two clusters in [31] to the semi-streaming model. We leave the technical details to the full version of the paper and, here, highlight the ingredients that make up the following result.

▶ **Theorem 7** (Formalization of Result 2). *There is an algorithm that, given a complete signed graph $G = (V, E^+ \cup E^-)$ and $\varepsilon \in (0,1)$, with high probability finds a partition (L, R) of V where*

$$\mathsf{frust}(G,\, L) \leq (1+\varepsilon) \cdot \mathsf{frust}(G).$$

Moreover, the algorithm uses $O\left(n \cdot \frac{\log^4(n/\varepsilon)}{\varepsilon^6}\right)$ space, a single pass over the stream and has running time of $O\left(\left(\frac{n}{\varepsilon}\right)^2 \cdot \log^3 n + n \log n \cdot \left(\frac{1}{\varepsilon}\right)^{O\left(\varepsilon^{-4}\right)}\right)$.

4.1 Evaluating Frustration

The first key ingredient is a tool for determining $\mathsf{frust}(G,\, L)$ for an arbitrary $L \subseteq V$, subject to the memory constraints of the semi-streaming setting. We turn to the idea of cut sparsifiers (see [4, 19, 42]), which store a sparse representation of G which approximately answers the cut queries for all $S \subseteq V$: what is the size of $E(S, V \setminus S)$?

▶ **Lemma 8.** *Let $G = (V, E^+ \cup E^-)$ be a complete signed graph. There is a single-pass streaming algorithm using $O\left(n \log^3 n/\varepsilon^2\right)$ space and $O\left(n^2 \log^2 n\right)$ time that, for any $\varepsilon \in (0,1)$, with high probability produces an oracle that, when given $L \subseteq V$, returns $\mathsf{frust}_\varepsilon(G,\, L)$ in $O\left(n \log n/\varepsilon^2\right)$ time where*

$$(1-\varepsilon)\mathsf{frust}(G,\, L) \leq \mathsf{frust}_\varepsilon(G,\, L) \leq (1+\varepsilon)\mathsf{frust}(G,\, L).$$

Our construction of the "frustration sparsifier" in Lemma 8 follows from a standard application of cut sparsifiers, and the fact that the portion of frustration contributed to by the "$-$" edges in each side can be estimated indirectly by looking at "$+$" edges crossing the cut.

Frustration Sparsifier for $G = (V, E^+ \cup E^-)$ and $\varepsilon \in (0,1)$:
- Let $H = (V, E_H, w_H)$ be an $\frac{\varepsilon}{2}$ cut sparsifier for $G' = (V, E^+)$.
- Return $\mathsf{frust}_\varepsilon(G, \cdot)$ which, when given $L \subseteq V$, computes

$$\underbrace{w_H(L, V \setminus L)}_{\approx |E^+(L, V\setminus L)|} + \underbrace{\left|E^-\right| - \left(|L| \cdot |V \setminus L| - w_H(L, V \setminus L)\right)}_{\approx |E^-(L)| + |E^-(V\setminus L)|}.$$

The proof of Lemma 8 can be found in the full version of this paper.

Observe as an aside that frustration sparsifiers alone give us a single-pass algorithm that uses $\widetilde{O}(n)$ space which, however, runs in exponential time: construct a frustration sparsifier with parameter ε, enumerate over all partitions of V, and return the one which gives the smallest estimated frustration.

4.2 A Polynomial Time Algorithm: Streamification of Giotis-Guruswami Algorithm

The next key ingredient is a simulation of the Giotis-Guruswami algorithm [31]. For the remainder of this section, let $\mathsf{frust}(G) = \gamma n^2$ where γ is not necessarily a constant. We are particularly interested in the case when $\gamma \leq \varepsilon/100^4$, which is the case of the Giotis-Guruswami algorithm which is non-trivial to simulate.

When there is no constraint on space usage, the Giotis-Guruswami algorithm works by sampling $\Theta(\log n)$ vertices (called the sample S) and trying every partition (S_L, S_R) of S. S_L and S_R are taken to be subsets of L and R respectively, where (L, R) is a partition of V. For each such partition (S_L, S_R), there should be a clear local choice for whether a vertex v should be assigned to L or to R by choosing the smaller of $\mathsf{Dis}(v, S_L + v, S_R)$[3] and $\mathsf{Dis}(v, S_L, S_R + v)$ respectively; we do this for all unsampled vertices, and refer to this step of the algorithm as MERGING. The algorithm then tries to further refine the solution quality by marking each vertex that improves the frustration when moved from L to R or vice versa, keeping every other vertex fixed. All marked vertices are then moved to the other part. The process of marking vertices and moving them after all have been marked is referred to as the SWITCHING step.

As it turns out, the correctness of the Giotis-Guruswami algorithm still holds with a small error introduced by graph sparsification. This is crucial as it allows us to design space efficient streaming algorithms by simply maintaining a cut sparsifier during the stream, and performing the Giotis-Guruswami procedure in the end. More concretely, only once the sparse representations of the input stream have been obtained (i.e. a single pass over the stream has been completed), do we then start iterating over every partition (S_L, S_R) of S, and in each iteration MERGING, SWITCHING, and storing the best resulting (L, R) seen so far. MERGING is easily ported to the streaming setting; we just need to store the $\Theta(n \log n)$ edges incident to S. The way SWITCHING is adapted, on the other hand, is less trivial; we store a sparse representation of the input graph by sampling $\Theta\big((1/\varepsilon^2)\log n\big)$ neighbors for each vertex. Finally, determining the (approximate) best partition seen through all iterations can be done by using a frustration sparsifier.

Streamification of [31] for $G = (V, E^+ \cup E^-)$ and $\varepsilon \in (0, 1)$, when $\gamma \lessapprox \varepsilon$:

1. Sample $100 \log n$ vertices uniformly at random. Call this set S.
2. Sample $100^3 \cdot \frac{\log n}{\varepsilon^2}$ vertices uniformly at random for each vertex v. Call these sets N_v.
3. Read through the stream of (signed) edges and
   - Store edges incident to S. Call this graph G_S.
   - Store edges joining v and N_v for all v. Call this graph G_N.
   - Store $\mathsf{frust}_{\varepsilon/10}(G, \cdot)$, a frustration sparsifier of G.
4. For every partition (S_L, S_R) of S

   (Merging)

   For every $v \in V \setminus S$, assign v to L if $\mathsf{Dis}_{G_S}(v, S_L + v, S_R) < \mathsf{Dis}_{G_S}(v, S_L, S_R + v)$, and R otherwise.

   S_L is assigned to L and S_R is assigned to R.

   (Switching)

   Mark $v \in L$ if $\mathsf{Dis}_{G_N}(v, L + v, R - v) > \mathsf{Dis}_{G_N}(v, L - v, R + v)$ and mark $v \in R$ similarly (flip the inequality).

   Switch assignments of all marked vertices from L to R or vice versa.

   Keep (L, R) if $\mathsf{frust}_{\varepsilon/10}(G, L)$ is the smallest seen so far.
5. Return (L, R).

[3] In this section, we use the following notation: For any set S and any vertex v, $S + v$ denotes $S \cup \{v\}$ and $S - v$ denotes $S \setminus \{v\}$.

The roles of MERGING and SWITCHING are roughly explained as follows. When (S_L, S_R) is congruent with how S would be partitioned in an optimal solution, MERGING assigns most vertices to L and R in the same way they would have been assigned in said optimal solution. These vertices do not change assignment when SWITCHING. The remaining vertices may have lots of disagreement with their current assignment in spite of having lots of agreement with S_L or S_R. SWITCHING rectifies this, minimizing the disagreement of these vertices with a large number of vertices (the ones that stay put, mentioned earlier).

We leave the proofs to the full version of the paper. Suffice to say, the core technical work here is in the analysis showing that weakly simulating SWITCHING in the way we have still yields a correct algorithm.

4.3 Looking at Exponentially Fewer Candidate Partitions

Observe that the first two ingredients yield an algorithm with $\Theta(n^{101})$ running time, where the inefficiency stems from having to iterate over all partitions of a sample of size $100 \log n$. The last key ingredient completes Theorem 7 by instead iterating over all partitions of a sample of size $100 \log \log n$, building from this a partition of size $O(\log n)$ via an independent sample (we call this step MINI-MERGING), and thereafter running MERGING and SWITCHING.

This gives us Theorem 7, and also Theorem 9 by running the algorithm in an offline setting regardless of it being a streaming algorithm.

▶ **Theorem 9** (Formalization of Result 3). *There is a randomized algorithm that, given a complete signed graph $G = (V, E^+ \cup E^-)$, and for any $\varepsilon \in (0, 1)$, with high probability returns a partition (L, R) of V where*

$$\mathsf{frust}(G,\, L) \leq (1 + \varepsilon) \cdot \mathsf{frust}(G).$$

Moreover, the algorithm has runtime $O\Big(\frac{n^2 \log^3 n}{\varepsilon^2} + n \log n \cdot (1/\varepsilon)^{O(\varepsilon^{-4})}\Big)$.

To the best of our knowledge, this is the first algorithm for Problem 2 and 2-correlation clustering that is nearly-linear in the input. A little more formalism is supplied in Appendix A, but much of the details are left to the full version of the paper.

5 Space Lower Bounds

In this section, we state space lower bounds that complement our algorithmic results.

- Any p-pass deterministic streaming algorithm solving Problem 1 requires $\Omega\Big(\frac{n}{p}\Big)$ space.
- Any single-pass randomized algorithm solving Problem 1 on a complete signed multi-graph, where edges are allowed to repeat twice, requires $\Omega(n)$ space.
- Any single-pass randomized streaming algorithm solving Problem 2 *exactly*, or that outputs the *exact* frustration index, requires $\Omega\big(n^2\big)$ space.

Combined with our upper bounds in Theorem 2 and Theorem 7, our results convey the following message: (i). It is possible to test structural balance with a very high space efficiency by taking advantage of the arrive-once inputs and randomness, and both aspects are *necessary*; and (ii). It is possible to achieve a $(1 + \varepsilon)$-approximation in $\tilde{O}(n)$ space and polynomial time for any constant ε. In contrast, the exact solution requires almost all the input to be stored.

5.1 Structural Balance Testing by Deterministic Algorithms

We first present space lower bound for deterministic streaming algorithms. The formal statement of the lower bound is as follows.

▶ **Proposition 10.** *Any deterministic p-pass algorithm for Problem 1 requires $\Omega\left(\frac{n}{p}\right)$ bits of space. In particular, any deterministic single-pass algorithm requires $\Omega(n)$ bits of space.*

The proof of our deterministic lower bound uses a reduction from EQUALITY, which is known to require $\Omega\left(\frac{n}{p}\right)$ bits of communication over p rounds. More can be found in the full version.

5.2 Structural Balance Testing by Randomized Algorithms with Multi-edges

The second lower bound is the most interesting from a technical standpoint. It shows that we need to use $\Omega(n)$ space if the inputs are complete signed graphs with multi-edges. We assume that the signs of the multi-edges between the same pair of vertices are *consistent*: if e_1 between (u, v) is "+", a second edge e_2 between (u, v) also has to be "+". The definition of structural balance still holds by checking one of the edges between each vertex pair.

The formal statement of our lower bound is as follows.

▶ **Proposition 11.** *Any single-pass streaming algorithm that correctly tests structural balance with probability at least $\frac{99}{100}$ on complete signed multi-graphs with sign-consistent multi-edges has to use a memory of $\Omega(n)$ bits.*

On the high level, the plan to prove Proposition 11 is a reduction from INDEX. The standard INDEX problem is described as follows: Alice holds a string $x \in \{0,1\}^N$, Bob holds an index $i^* \in [N]$, and the two players want to learn the value of x_{i^*}.

In a first attempt at a reduction, the two players are initially given $2N + 2$ vertices, with each bit x_j corresponding to two vertices u_j and v_j and two special vertices s and t. Alice creates two empty sets A and B, and for each input bit x_j, $j \in [N]$, Alice puts u_j to A and v_j to B if $x_j = 0$, and v_j to A and u_j to B if $x_j = 1$. Then, for every vertex except u_{i^*} and v_{i^*}[4], Alice adds "+" edges from s to the vertices in A and from t to vertices in B. Furthermore, for each pair of vertices *both inside* A or B, Alice adds a "+" edge; and for vertex pairs such that $u \in A$ and $v \in B$, Alice adds a "−" edge. On the other end, Bob completes the graph by adding "+" edges to (s, u_{i^*}) and (t, v_{i^*}) and "−" edges to (s, v_{i^*}) and (t, u_{i^*}). Now, if $x_{i^*} = 0$, the resulting graph is balanced; otherwise, the graph is unbalanced since v_{i^*} is in A.

While the plan sounds nice, it does *not* work. Careful readers may have already found the problem: Alice does *not* know which vertices correspond to u_{i^*} and v_{i^*}; and if she somehow infers it, the problem becomes easy as she can output x_{i^*} without any communication. As such, for the above idea to work, we need to somehow "hide" the index to the holder of the string, but give them the ability to create graphs that leave everything but x_{i^*}.

To the above end, we introduce the following version of Leave-One INDEX.

▶ **Problem 3** (Leave-One INDEX)**.** *Consider a communication problem between 3 players, namely Alice, Bob, and Charlie. Both Alice and Bob hold the same string $x \in \{0,1\}^N$, and Charlie holds an index $i^* \in [N]$. Furthermore, Alice holds $S \subset [N]$ and Bob holds $T \subset [N]$, such that $S \cup T = [N] \setminus \{i^*\}$. The communication goes in the order of Alice, Bob and Charlie, and the players want to output x_{i^*}.*

[4] The construction is in fact not possible as we will see in the next paragraph; it is here to explain the intuition.

The sets of S and T create some "flexibility" for Alice and Bob to infer the index. Indeed, if $S \cap T = \emptyset$, the problem becomes easy as Bob can learn Alice's bits from a message that simply encodes which bits are missing[5]. Nevertheless, we prove (see Appendix B) that the worst-case communication complexity of the Leave-One INDEX problem is still $\Omega(N)$. The formal lower bound is as follows.

▶ **Lemma 12.** *Any (possibly randomized) one-way communication protocol for the Leave-One INDEX defined in Problem 3 that outputs x_{i^*} correctly with probability at least $\frac{99}{100}$ requires $\Omega(N)$ bits of communication.*

Figure 1 shows how Alice, Bob, and Charlie can use the previously mentioned (wrong) idea to get a (now) correct reduction of Leave-One Index to testing structural balance, where the multi-edges occur from the intersection $S \cap T$ of Alice's and Bob's sets.

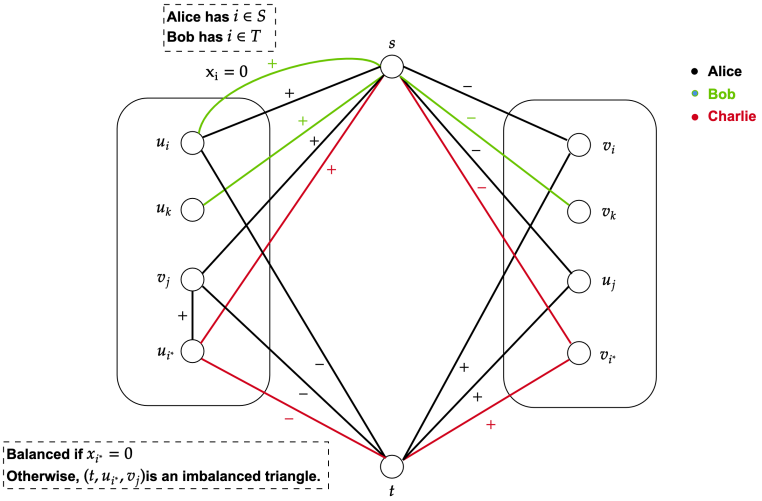

Figure 1 An illustration of the reduction from Leave-One Index.

5.3 Frustration-minimizing Partition by Exact Algorithms

Finally, we show that any single-pass streaming algorithm that solves the frustration index *exactly* has to store almost the entire graph.

▶ **Proposition 13.** *Any single-pass streaming algorithm that solves Problem 2 exactly or returns the optimal frustration index value requires $\Omega(n^2)$ bits of space.*

Proposition 13 goes through communication complexity again. This time, our reduction is from INDEX (a matrix variant thereof) which is hard under the one-way communication regime. Details are to be found in the full version.

[5] In particular, there is a simple protocol with 1 bit of communication when $S \cap T = \emptyset$: Alice takes XOR of all bits in $x[S]$, and send it to Bob. Bob takes XOR of all bits in $x[T]$ he will learn the XOR of $x[S \cup T]$. Bob then compares the XOR of x vs. the XOR of all bits in $x[S \cup T]$ to learn x_{i^*}

References

1. Robert P. Abelson and Milton J. Rosenberg. Symbolic psycho-logic: A model of attitudinal cognition. *Behavioral Science*, 3(1):1–13, 1958.
2. Amit Agarwal, Moses Charikar, Konstantin Makarychev, and Yury Makarychev. $O(\sqrt{\log n})$ approximation algorithms for min UnCut, min 2CNF deletion, and directed cut problems. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing – STOC '05*, New York, New York, USA, 2005. ACM Press.
3. Kook Jin Ahn, Graham Cormode, Sudipto Guha, Andrew McGregor, and Anthony Wirth. Correlation clustering in data streams. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 2237–2246. JMLR.org, 2015.
4. Kook Jin Ahn and Sudipto Guha. Graph sparsification in the semi-streaming model. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris E. Nikoletseas, and Wolfgang Thomas, editors, *Automata, Languages and Programming, 36th Internatilonal Colloquium, ICALP 2009*, volume 5556 of *Lecture Notes in Computer Science*, pages 328–338. Springer, 2009. doi:10.1007/978-3-642-02930-1_27.
5. Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating inconsistent information: ranking and clustering. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 684–693. ACM, 2005.
6. Claudio Altafini. Dynamics of opinion forming in structurally balanced social networks. *PloS one*, 7(6):e38135, 2012.
7. T Antal, P L Krapivsky, and S Redner. Social balance on networks: The dynamics of friendship and enmity, 2006.
8. Samin Aref, Andrew J Mason, and Mark C Wilson. A modeling and computational study of the frustration index in signed networks. *Networks*, 75(1):95–110, January 2020.
9. Sepehr Assadi, Yu Chen, and Sanjeev Khanna. Polynomial pass lower bounds for graph streaming algorithms. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019.*, pages 265–276, 2019.
10. Sepehr Assadi and Ran Raz. Near-quadratic lower bounds for two-pass graph streaming algorithms. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*, pages 342–353. IEEE, 2020. doi:10.1109/FOCS46700.2020.00040.
11. Sepehr Assadi and Chen Wang. Sublinear Time and Space Algorithms for Correlation Clustering via Sparse-Dense Decompositions. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*, volume 215 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 10:1–10:20, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ITCS.2022.10.
12. Adi Avidor and Michael Langberg. The multi-multiway cut problem. *Theor. Comput. Sci.*, 377(1):35–42, May 2007.
13. Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Mach. Learn.*, 56(1-3):89–113, 2004. doi:10.1023/B:MACH.0000033116.57574.95.
14. Francisco Barahona. On the computational complexity of Ising spin glass models. *Journal of Physics A: Mathematical and General*, 15(10):3241–3253, 1982.
15. MohammadHossein Bateni, Hossein Esfandiari, and Vahab S. Mirrokni. Almost optimal streaming algorithms for coverage problems. In *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2017*, pages 13–23, 2017.
16. Anubhav Baweja, Justin Jia, and David P. Woodruff. An efficient semi-streaming PTAS for tournament feedback arc set with few passes. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference, ITCS 2022*, volume 215 of *LIPIcs*, pages 16:1–16:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

17 Soheil Behnezhad, Moses Charikar, Weiyun Ma, and Li-Yang Tan. Almost 3-approximate correlation clustering in constant rounds. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022*, pages 720–731. IEEE, 2022. doi:10.1109/FOCS54457.2022.00074.

18 Soheil Behnezhad, Moses Charikar, Weiyun Ma, and Li-Yang Tan. Single-pass streaming algorithms for correlation clustering. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023*, pages 819–849. SIAM, 2023. doi:10.1137/1.9781611977554.ch33.

19 András A. Benczúr and David R. Karger. Approximating s-t minimum cuts in $\tilde{O}(n^2)$ time. In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 47–55. ACM, 1996. doi:10.1145/237814.237827.

20 Andrej Bogdanov and Emanuele Viola. Pseudorandom bits for polynomials. *SIAM J. Comput.*, 39(6):2464–2486, 2010. doi:10.1137/070712109.

21 Dorwin Cartwright and Frank Harary. Structural balance: a generalization of Heider's theory. *Psychol. Rev.*, 63(5):277–293, September 1956.

22 Amit Chakrabarti, Prantar Ghosh, Andrew McGregor, and Sofya Vorotnikova. Vertex ordering problems in directed graph streams. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020*, pages 1786–1802, 2020.

23 Sayak Chakrabarty and Konstantin Makarychev. Single-pass pivot algorithm for correlation clustering. keep it simple! *arXiv preprint arXiv:2305.13560*, 2023.

24 Moses Charikar, Venkatesan Guruswami, and Anthony Wirth. Clustering with qualitative information. In *Proceedings of the 44th Symposium on Foundations of Computer Science (FOCS 2003)*, pages 524–533, 2003.

25 Shuchi Chawla, Konstantin Makarychev, Tselil Schramm, and Grigory Yaroslavtsev. Near optimal LP rounding algorithm for correlation clustering on complete and complete k-partite graphs. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015*, pages 219–228. ACM, 2015.

26 Flavio Chierichetti, Nilesh N. Dalvi, and Ravi Kumar. Correlation clustering in mapreduce. In Sofus A. Macskassy, Claudia Perlich, Jure Leskovec, Wei Wang, and Rayid Ghani, editors, *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14*, pages 641–650. ACM, 2014.

27 Vincent Cohen-Addad, Silvio Lattanzi, Slobodan Mitrovic, Ashkan Norouzi-Fard, Nikos Parotsidis, and Jakub Tarnawski. Correlation clustering in constant many parallel rounds. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021*, volume 139 of *Proceedings of Machine Learning Research*, pages 2069–2078. PMLR, 2021. URL: http://proceedings.mlr.press/v139/cohen-addad21b.html.

28 Vincent Cohen-Addad, Euiwoong Lee, and Alantha Newman. Correlation clustering with Sherali-Adams. *CoRR*, abs/2207.10889, 2022.

29 Bhaskar DasGupta, German Andres Enciso, Eduardo Sontag, and Yi Zhang. Algorithmic and complexity results for decompositions of biological networks into monotone subsystems. *Biosystems.*, 90(1):161–178, July 2007.

30 James A Davis. Clustering and structural balance in graphs. *Human relations*, 20(2):181–187, 1967.

31 Ioannis Giotis and Venkatesan Guruswami. Correlation clustering with a fixed number of clusters. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, SODA '06, pages 1167–1176, 2006.

32 Frank Harary. On the notion of balance of a signed graph. *Michigan Math. J.*, 2(2), January 1953.

33 Frank Harary. Signed graphs for portfolio analysis in risk management. *IMA Journal of Management Mathematics*, 13(3):201–210, 2002.

34 Frank Harary. On the measurement of structural balance. *Behavioral Science*, 4(4):316–323, 2007.

35 Fritz Heider. Attitudes and cognitive organization. *J. Psychol.*, 21:107–112, January 1946.

36 Fritz Heider. *The Psychology of Interpersonal Relations*. Psychology Press, 1982.

37 Falk Hüffner, Nadja Betzler, and Rolf Niedermeier. Separator-based data reduction for signed graph balancing. *J. Comb. Optim.*, 20(4):335–360, November 2010.

38 Giovanni Iacono, Fahimeh Ramezani, Nicola Soranzo, and Claudio Altafini. Determining the distance to monotonicity of a biological network: a graph-theoretical approach. *IET Syst. Biol.*, 4(3):223–235, May 2010.

39 Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, STOC '02, pages 767–775, New York, NY, USA, May 2002. Association for Computing Machinery.

40 Funda Kivran-Swaine, Priya Govindan, and Mor Naaman. The impact of network structure on breaking ties in online social networks: unfollowing on twitter. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1101–1104. ACM, 2011.

41 Shachar Lovett. Unconditional pseudorandom generators for low degree polynomials. *Theory Comput.*, 5(1):69–82, 2009. doi:10.4086/toc.2009.v005a003.

42 Andrew McGregor. Graph stream algorithms: a survey. *SIGMOD Rec.*, 43(1):9–20, 2014. doi:10.1145/2627692.2627694.

43 Michael Moore. An international application of Heider's balance theory. *Eur. J. Soc. Psychol.*, 8(3):401–405, July 1978.

44 Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM J. Comput.*, 22(4):838–856, 1993. doi:10.1137/0222053.

45 Christos Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, STOC '88, pages 229–234, New York, NY, USA, January 1988.

46 Ron M. Roth. *Introduction to coding theory*. Cambridge University Press, 2006.

47 Christopher Sibona. Unfriending on facebook: Context collapse and unfriending behaviors. In *2014 47th Hawaii International Conference on System Sciences*, pages 1676–1685. ieeexplore.ieee.org, January 2014.

48 Jiliang Tang, Yi Chang, Charu Aggarwal, and Huan Liu. A survey of signed network mining in social media. *ACM Computing Surveys (CSUR)*, 49(3):1–37, 2016.

49 Andreia Sofia Teixeira, Francisco C Santos, and Alexandre P Francisco. Emergence of social balance in signed networks. In *Complex Networks VIII*, pages 185–192. Springer International Publishing, 2017.

50 Haotian Wang, Feng Luo, and Jie Gao. Co-evolution of opinion and social tie dynamics towards structural balance. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA'22)*, pages 3362–3388, January 2022.

51 Bo Xu, Yun Huang, Haewoon Kwak, and Noshir Contractor. Structures of broken ties: Exploring unfollow behavior on twitter. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, CSCW '13, pages 871–876, 2013. doi:10.1145/2441776.2441875.

52 Mihalis Yannakakis. Edge-Deletion problems. *SIAM J. Comput.*, 10(2):297–309, May 1981.

53 Thomas Zaslávsky. Balanced decompositions of a signed graph. *J. Combin. Theory Ser. B*, 43(1):1–13, August 1987.

A Offline Solution to Problem 2 (More Details)

Here we present an offline algorithm for solving Problem 2 when $\gamma < \varepsilon/100^4$ (which is an important part leading to Theorem 9), which can be transferred to the semi-streaming setting with superficial modification.

Nearly-Linear Time EPTAS solving Problem 2 for $G = (V, E^+ \cup E^-)$ and $\varepsilon \in (0, 1)$, when $\gamma < \varepsilon/100^4$:

1. Sample $2000 \log n$ vertices uniformly at random from V. Call this set S and let $G_S = (V, E(S) \cup E(S, V \setminus S))$.
2. Sample $100 \log \log n$ vertices uniformly at random from S. Call this set S', the *seed-set*, and let $G_{S'} = (V, E(S') \cup E(S', V \setminus S'))$.
3. Sample $100^3 \cdot \frac{\log n}{\varepsilon^2}$ vertices uniformly at random for each vertex v. Call these sets N_v and let $G_N = (V, \bigcup_v E(v, N_v))$.
4. Construct $\mathsf{frust}_{\varepsilon/10}(G, \cdot)$, a frustration sparsifier of G.
5. For every partition (S'_L, S'_R) of S'
   (Mini-merging on S')
   For every $v \in S \setminus S'$, assign v to S_L if $\mathsf{Dis}_{G_{S'}}(v, S'_L + v, S'_R) < \mathsf{Dis}_{G_{S'}}(v, S'_L, S'_R + v)$, and S_R otherwise.
   S'_L is assigned to S_L and S'_R is assigned to S_R.
   (Merging on S)
   For every $v \in V \setminus S$, assign v to L if $\mathsf{Dis}_{G_S}(v, S_L + v, S_R) < \mathsf{Dis}_{G_S}(v, S_L, S_R + v)$, and R otherwise.
   S_L is assigned to L and S_R is assigned to R.
   (Switching)
   Mark $v \in L$ if $\mathsf{Dis}_{G_N}(v, L + v, R - v) > \mathsf{Dis}_{G_N}(v, L - v, R + v)$ and mark $v \in R$ similarly (flip the inequality).
   Switch assignments of all marked vertices from L to R or vice versa.
   Keep (L, R) if $\mathsf{frust}_{\varepsilon/10}(G, L)$ is the smallest seen so far.
6. Return (L, R).

B Proof of Lemma 12

In the proof of the communication complexity for the standard INDEX problem, the intuition is that the first player (Alice) has no knowledge of i^* other than knowing it is uniformly distributed (in the particular hard distribution). As such, the problem can only be solved if a sufficiently large number of bits are communicated. However, in the Leave-One INDEX problem (Problem 3), the first two players (Alice and Bob) might be able to learn where exactly i^* is by comparing their sets of indices S and T. In Problem 3, by the promise that $i^* \notin T$, Bob can learn the distribution of i^* better than uniform even without communication.

Our plan to prove Lemma 12 is to conduct a case analysis on whether the communication between Alice and Bob somehow "solves" the problem. To be precise, we separate the cases based on whether Alice's message significantly changes the distribution of i^* from Bob's perspective. If such a case happens, then we can use a result from [10] to show that the communication between *Alice and Bob* has to be $\Omega(\varepsilon^2 N)$. On the other hand, if the message from Alice does *not* tell Bob a lot about the distribution of i^*, a large number of bits are still required for the communication between *Bob and Charlie* since there are constant fraction of coordinates where i^* still looks almost uniform. The latter bound cannot be obtained directly by a reduction since i^* is already not uniform from Bob's perspective. Nevertheless, we can adapt the information-theoretic proof for INDEX to get the desired lower bound in a standard manner.

We now formalize the above intuition. Let $\Pi_{A,B}$ be the random variable for the message between Alice and Bob, and $\Pi_{B,C}$ be the random variable for the message between Bob and Charlie. Furthermore, let X be the random variable for the string being held by Alice and Bob, and let I be the random variable for i^*. For the choice of the set T, we use T' to denote the realization. Conditioning on the first message $\Pi_{A,B} = \pi_{A,B}$ and Bob's local inputs X, T, we let $I^B := I \mid X, T, \Pi_{A,B}$ be the random variable for i^* from *Bob's internal perspective after message* $\pi_{A,B}$. We slightly abuse notation and let these random variables also represent their distributions.

As promised, the first case we are going to show is the communication lower bound between Alice and Bob if the distribution of i^* that Bob learns is ε-far from his *prior* knowledge. Formally, we define the notation of ε-learn as follows.

▶ **Definition 14.** *Let $\pi_{A,B}$ be the message of a randomized one-way communication protocol. We say that Bob ε-learns i^* if after the message $\pi_{A,B}$, in expectation, the distribution of i^* from Bob's perspective is more than ε-far from the distribution he knows from his own input in the total variation distance, i.e.*

$$\mathbb{E}[\|I \mid \Pi_{A,B}, X, T - I \mid X, T\|_{\mathrm{TVD}}] > \varepsilon,$$

where the expectation is over the randomness of X, T and $\Pi_{A,B}$.

The notion of ε-learn was first developed by [9,10]. As such, we can prove a communication lower bound for Bob to ε-learn i^*.

▶ **Lemma 15.** *Any such protocol $\pi_{A,B}$ for Bob to ε-learn i^* has to use $\Omega(\varepsilon^2 \cdot N)$ bits of communication between Alice and Bob.*

Proof. We prove the lemma by a direct reduction from [10], which address the set intersection problem. The definition of the problem and the communication complexity is as follows.

▶ **Proposition 16** ([10])**.** *The Set-Intersection is a two-player game between Alice and Bob. The two players are given $A \subseteq [m]$ and $B \subseteq [m]$, respectively, with the promise that there exists a unique element e^* such that $e^* = A \cap B$. The goal is to find the target element e^* after one-way communication. Let E be the distribution of e^*, it is known that any communication protocol that achieves*

$$\mathbb{E}[\|E \mid \Pi, B - E \mid B\|_{\mathrm{TVD}}] > \varepsilon$$

has to use a communication of $\Omega(\varepsilon^2 m)$ bits, where the expectation is over the randomness of B and the protocol's randomness Π. In particular, the communication lower bound applies to the following distribution.

A hard distribution for Set-Intersection.

- *Sample two* disjoint *sets of coordinates A' and B' of size $N/4 - 1$ each uniformly at random from $[N]$.*
- *Sample an element e^* uniformly at random from $[N] \setminus (A' \cup B')$, and let $A := A' \cup \{e^*\}$, $B := B' \cup \{e^*\}$.*

The result of Proposition 16 can be generalized to back-and-forth communication if we allow Alice *or* Bob to ε-learn the distribution. However, for the purpose of our reduction, the one-way version suffices. The reduction from Set-Intersection to the communication between Alice and Bob in Lemma 12 is as follows.

Inputs: Alice holds $A \subseteq [m]$, Bob holds $B \subseteq [m]$, $A \cap B = e^*$.

A communication protocol PROT for Lemma 12 that ε-learns the index i^* as in Lemma 15.

1. Alice creates a set $S' = [m] \setminus A$.
2. Bob creates a set $T' = [m] \setminus B$.
3. Alice runs PROT, sends to Bob. Bob uses the distribution for i^* as the distribution for e^*.

To prove the correctness of the reduction, we only need to show that a). S' and T' are valid inputs for PROT and b). $i^* = e^*$. It is straightforward to verify these conditions: we know that an element is *not* covered by $S' \cup T'$ if and only if *both* A and B cover it, which is exactly e^*. Therefore, the desired lower bound is established. ◀

Lemma 15 implies that with $o(N)$ bits of communication it is impossible for Bob to gain knowledge of the distribution of I that is significantly different from what he already knows. We now observe that with the input X and T, and conditioning on Alice's message does *not* significantly change Bob's distribution, Bob's distribution of I is close to uniform on the $[N] \setminus T$ coordinates.

▷ Claim 17. Conditioning on the event that Bob does *not* ε-learn i^*, the distribution of i^* that Bob learns from T and Alice's message $\pi_{A,B}$ is at most ε-far from uniform in expectation over the supports of $[N] \setminus T$, i.e.,

$$\mathop{\mathbb{E}}_{T,\Pi_{A,B}} [\|I \mid T, \Pi_{A,B} - U([N] \setminus T)\|_{\text{TVD}}] \leq \varepsilon.$$

Proof. We first show that without Alice's message, the distribution I of Bob restricted to $[N] \setminus T$ is uniform. This is a simple observation: with the promise that $i^* \notin T$, Bob can safely discard all coordinates therein. Since the string x is *independent* of i^*, Bob does not learn anything from x. As such, Bob's distribution remains *uniform* on the $[N] \setminus T$ coordinates.

We now condition on the fact that Bob does not ε learn i^*. As such, there is

$$\begin{aligned}
&\mathop{\mathbb{E}}_{X,T,\Pi_{A,B}} [\|I \mid X, T, \Pi_{A,B} - U([N] \setminus T)\|_{\text{TVD}}] \\
&\leq \mathop{\mathbb{E}}_{X,T,\Pi_{A,B}} \big[\|I \mid X, T - U([N] \setminus T)\|_{\text{TVD}} + \|I^B - I \mid X, T\|_{\text{TVD}}\big] \qquad \text{(triangle inequality)} \\
&\leq \mathop{\mathbb{E}}_{X,T,\Pi_{A,B}} [0 + \|I \mid \Pi_{A,B}, X, T - I \mid X, T\|_{\text{TVD}}] \leq \varepsilon,
\end{aligned}$$

where the last inequality directly comes from our assumption of no ε-learning.

Finally, note that the choice of i^* is independent of X (whether or not conditioning on $\Pi_{A,B}$). Therefore, we have

$$\begin{aligned}
&\mathop{\mathbb{E}}_{X,T,\Pi_{A,B}} [\|I \mid X, T, \Pi_{A,B} - U([N] \setminus T)\|_{\text{TVD}}] \\
&= \mathop{\mathbb{E}}_{T,\Pi_{A,B}} [\|I \mid T, \Pi_{A,B} - U([N] \setminus T)\|_{\text{TVD}}] \\
&\leq \varepsilon,
\end{aligned}$$

as desired. ◁

We now use the result of Claim 17 to prove the communication complexity conditioning on Bob does not ε-learn i^*. To continue, we insist that the distribution T where T' is sampled from always produces size-t sets. As such, we can establish the following lemma.

▶ **Lemma 18.** *Suppose the players sample from a distribution T to ensure the size of the set given to Bob is always t. Conditioning on the event that Bob does* not *ε-learn i^*, any communication protocol such that Charlie correctly outputs x_{i^*} with probability at least $1-\delta$ requires at least $(1-H_2(\delta)-2\varepsilon)\cdot(N-t)$ bits of communication.*

On a high level, the proof of Lemma 18 uses a strategy that is similar to the information-theoretic proof of INDEX, but it controls the entropy on the coordinates outside T. The sketch of the proof is as follows. By Fano's inequality, for Charlie to output x_{i^*} correctly with probability at least $1-\delta$, there has to be $\mathbb{H}(X_I \mid \Pi_{A,B}, \Pi_{B,C}, T, I) \leq H_2(\delta)$. Using Claim 17, we can prove that if Bob does *not* ε-learn i^*, the "uncertainty" of X_I – measured by the entropy – is still comparable to the entropy of X_I restricted to the $[N]\setminus T$ coordinates, i.e.,

$$\mathbb{H}(X_I \mid \Pi_{A,B}, \Pi_{B,C}, T, I) \geq \frac{1}{N-t}\cdot \mathbb{H}\big(X_{[N]\setminus T} \mid \Pi_{A,B}, \Pi_{B,C}, T\big) - 2\varepsilon.$$

Therefore, we can apply Fano's inequality to lower bound the mutual information between $X_{[N]\setminus T}$ and the information revealed by the protocol, i.e., $(\Pi_{A,B}, \Pi_{B,C})$, which in turn gives us the lower bound on the number of bits to communicate. We leave the proof of Lemma 18 to the full version.

Finalizing the proof of Lemma 12. For completeness, we construct the hard distribution of Problem 3 as follows.

A hard distribution for Problem 3.
1. Sample A and B according to the hard distribution for Set-Intersection prescribed in Proposition 16.
2. Let $S = [N]\setminus A$, and $T = [N]\setminus B$. Sample X from $\{0,1\}^N$ uniformly at random.
3. Give (X, S) to Alice, (X, T) to Bob, and i^* to Charlie.

Observe that in the above distribution, for any choice of T', there is $t = |T'| = \frac{3}{4}\cdot N$. Furthermore, it agrees with the hard distribution of Proposition 16 through the reduction in Lemma 15.

If Bob ε-learns i^* for $\varepsilon = \frac{1}{100}$, then the communication lower bound is already $\Omega\big(N^2\big)$ (by Lemma 15). Therefore, we condition on the case when Bob does *not* (1/100)-learns i^*. As such, let us conditioning on the event that $\mathbb{E}\big[\|I^B - I \mid X, T\|_{\text{TVD}}\big] \leq \frac{1}{100}$; and by Lemma 18, to make up the success probability of $\frac{99}{100}$, the protocol needs to send a message of length at least $(1-H_2(\frac{1}{100})-2/100)\cdot(N-t) = \Omega(N)$ bits, as desired. ◀