Comparison Graphs: A Unified Method for Uniformity Testing

Uri Meir
Tel Aviv University, Israel

Abstract

Distribution testing can be described as follows: q samples are being drawn from some unknown distribution P over a known domain $[n]$. After the sampling process, a decision must be made about whether P holds some property, or is far from it. The most studied problem in the field is arguably *uniformity testing*, where one needs to distinguish the case that P is uniform over $[n]$ from the case that P is ϵ-far from being uniform (in ℓ_1). It is known that for this task $\Theta\left(\sqrt{n}/\epsilon^2\right)$ samples are necessary and sufficient. This problem was recently considered in various restricted models that pose, for example, communication or memory constraints. In more than one occasion, the known optimal solution boils down to counting collisions among the drawn samples (each two samples that have the same value add one to the count). This idea dates back to the first uniformity tester, and was coined the name "collision-based tester".

In this paper, we introduce the notion of *comparison graphs* and use it to formally define a generalized collision-based tester. Roughly speaking, the edges of the graph indicate the tester which pairs of samples should be compared (that is, the original tester is induced by a clique, where all pairs are being compared). We prove a structural theorem that gives a sufficient condition for a comparison graph to induce a good uniformity tester. As an application, we develop a generic method to test uniformity, and devise nearly-optimal uniformity testers under various computational constraints. We improve and simplify a few known results, and introduce a new constrained model in which the method also produces an efficient tester.

The idea behind our method is to translate computational constraints of a certain model to ones on the comparison graph, which paves the way to finding a good graph: a set of comparisons allowed by the model that suffice to test for uniformity. We believe that in future consideration of uniformity testing in new models, our method can be used to obtain efficient testers with minimal effort.

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

Keywords and phrases Distribution Testing, Uniformity Testing, Distributed Algorithms, Streaming Algorithms, Comparison Graphs

Digital Object Identifier 10.4230/LIPIcs.ITCS.2021.17

Related Version A full version of the paper is available at `https://arxiv.org/abs/2012.01882`.

Acknowledgements The author would like to thank Rotem Oshman for valuable counsel, Noam Mazor for very helpful discussions, and Ami Paz as well as the anonymous referees of ITCS 2021 for their useful comments.

1 Introduction

The field of property testing was initiated by [10, 25, 20] and concerns with fast probabilistic algorithms that use query access to some large structure (such as graphs, functions, distribution, etc.) in order to determines whether a specific instance belong to a subclass of possible instances (e.g., connected graphs or monotone functions) or in some sense far from it. Specifically for distributions, as formulated in [9], we are given random samples from some unknown distribution, and we wish to decide with high probability (over the random samples) whether it has some property, or it is far from any distribution that does (typically we use ℓ_1 as a distance measure). Properties of distributions were excessively studied through the years (see [18, 11] for excellent surveys). Until recently, the vast majority of results were limited to the classic setting, where a *single* processor is given an oracle access, and performs the testing procedure. The measure of complexity is based solely on the number of samples, as typically the running time is polynomial in this number.

However, distribution testing can be very useful in different frameworks as well. For example, suppose we have a sensor network taking some measurements that need to be combined in order to make a decision about the subject of these measurements, be it volcanic activity, seismic movements, or any form of radiation.

Another framework where testing is useful, is under constrained memory. One might try testing an object so big, that the number of samples needed is very large. In that scenario, even if one might endure a lengthly sampling process, storing all past samples at one given moment can be too costly. For example, imagine a large telescope collecting data on infrared radiation in a pursuit to discover new planets. These types of questions can be translated to a *streaming model*, where samples come as a stream, and one wish to store only a small amount of data while reliably test for a property of the underlying distribution.

When considering the relatively similar motivations for these cases, one might even wonder about a combination of the two models, where multiple sensors spread out in some area collect samples, each of which has bounded memory. They will then need to process all available data within their memory constraints, such that by the end of each time period - they are able to report their individual findings concisely to some data center that aggregates all information in order to make an important decision.

All of these questions are inherently multi-dimensional, in the sense that many incomparable resources are in play: the number of players, the number of samples, the memory space of each sensor, and even the amount of bits communicated. Here, we focus our efforts, for the most part, on minimizing the *sample complexity* (or amount of samples per player, when multiple players are involved), with other resources given as parameters.

This specification is well-motivated by the case where we have no shortage of data we can sample from, and we wish to understand how long a sampling process should take, as a parameter of the number of sensors we use and their computational strength. Throughout the text, we focus on the task of uniformity testing, a key problem in the field of distribution testing.

Uniformity testing. The most studied family of problems in distribution testing is arguably *identity testing*, where we want to test whether the input distribution P is equal to some fixed distribution p, or ϵ-far from it (in ℓ_1), where ϵ is the proximity parameter of the problem. In the heart of these problems stands the problem of *uniformity testing*, where $p = U_\Omega$. One evidence for the importance of uniformity testing was shown in [15] and made more robust in [19]: it is actually complete for identity testing, in the sense that testing identity to any fixed distribution p can be reduced to testing uniformity instead. On the other hand, uniformity testing is a specific case of other problems such as closeness testing and independence testing, so showing lower bounds for uniformity testing would imply lower bounds for these problems. The classic version of testing uniformity was settled for the case $\epsilon = \Omega\left(n^{-1/4}\right)$ in [24], showing that $\Theta\left(\sqrt{n}/\epsilon^2\right)$ samples are in fact sufficient and necessary. [26] later showed this holds for all values of ϵ.

Collision-based testers. The problem of uniformity testing was implicitly introduced in [21], as a way to test the expansion of a graph: when simulating multiple short random walks, and observing the distribution of the endpoint, one can connect a uniform distribution over these endpoint to good expansion of the graph. To solve uniformity, the *collision-based* tester was introduced, where one simply counts the number of pairs of samples that have the same value. This tester was shown to have sample complexity of $\Theta\left(\sqrt{n} \cdot \mathrm{poly}(1/\epsilon)\right)$, which is turned out to be sub-optimal in terms of ϵ. However, several years after the question was settled, it was shown in [14] that the original collision-based tester also achieves optimal sample complexity, using a finer analysis.

The idea behind the collision-based tester is rather straightforward: when comparing two samples from a distribution, the chance of both having the same value (also referred as *collision probability*) relates to the ℓ_2 norm of the distribution. It is a well-known fact that over a fixed set Ω, the uniform distribution has the minimal ℓ_2 norm. It is also rather easy to show that any distribution that is somewhat far from uniform (in statistical distance), has a significantly larger ℓ_2 norm. This means that each comparison of two samples is an unbiased estimator of the collision probability (having the right expectation), but with very high variance. One would need to average over many such comparisons in order to reduce the variance.

Other methods to test for uniformity. Over the years, numerous methods to test for uniformity have been proposed. Some of them had different goals in mind, such as testing with very high confidence, or in a multiparty model where each player gets a single sample (sometimes even wishing to keep it private). These methods include counting unique element [24], modified χ^2 test [26], using the empirical distance to uniformity [13], randomly hashing samples to a smaller domain [6] and more. In both [12, 16], testers that aim to overcome different constraints relied strongly on collision counting[1]. Considering it is also optimal in the classic setting, this makes collision-based testing a prime candidate for a more generic method to test uniformity, and hopefully adjust itself to different models easily.

Comparison graphs. The original version of the collision-based tester takes a set of samples, and use comparisons between *all pairs of samples*. This is well-suited for the classic model, where one processor sees all the samples, and can easily perform all comparisons.

However, in more constrained models, this simple task is inherently impossible. For example, a memory-constrained tester cannot store all previous samples in order to compare them with new ones. In the simultaneous model, where each processor holds its own set of samples, a lot of communication might be needed to compare samples that are held by different processors. In distributed models, such as CONGEST and LOCAL (see [16]), the problem is defined where each player in a network holds a single sample from a distribution (replacing one sample by a constant amount produces similar behaviour). In these models, it is much cheaper for player to compare their samples with those of a neighboring player, than it is to make such a comparison with players that are far away (on the network topology).

To this end, we introduce the notion of *comparison graphs*. A comparison graph is linked to a collision-based tester (or algorithm) as follows: the vertices of the graph are the samples given as input, and the edges are pairs of samples that are being compared. As stated above: in the classic model this graph is typically the complete clique (all pairs of samples are compared). Under constrained models, however, very specific edges (comparisons) are allowed, whereas others are not. For example, if the sample s_1 is given to one player, and the sample s_2 is given to another player in the simultaneous model, no algorithm can presume to compare the two samples.

[1] These testers do not count collisions per se, they use additional crucial steps.

Equipped with the notion of comparison graphs, one can define a *collision-based* tester as a couple (G, τ), where $G = (V, E)$ is the comparison graph that defines which comparisons are being made, and τ is a threshold parameter. The algorithm is defined as follows: it counts the amount of collisions observed, Z, and compares it to a threshold T that depends on τ and the amount of comparisons made (which is $|E|$).

Reliable collisions-based testers In this work we introduce a structural theorem concerning with which sets of comparisons are able to inspire a reliable test for uniformity based solely on counting collisions. This is done by observing the comparison graph G. We formulate sufficient conditions in terms of the graph G, that guarantee it induces a good tester (when paired with the right threshold parameter τ). It turns out that two properties of a comparison graph G are key: the first is the number of edges, which represents the amount of comparisons being made; the second one, somewhat surprisingly, is the number of 2-paths in the graph G, which encapsulates the amount of dependencies between different comparisons being made.

Few of our testers rely on the same type of graph, that pops up multiple times, for different reasons. To this end, we formulate Lemma 4, that specifies the required parameters for a comparison graph of this type to induce a good tester.

1.1 Examples of comparison graphs

It is interesting that the number of samples (the measure we usually wish to minimize) does *not* appear as a condition on our comparison graph directly. However, it does play a role indirectly, as simple inequalities connect the three graph quantities (see Section 4.1).

Our structural theorem basically shows that any comparison graph with enough edges, but not-too-many 2-paths induces a good uniformity tester. To better understand the meaning of this, we fix the amount of edges, $|E|$, and review a short list of examples for potential comparison graphs. We are interested in the interplay between the amount of vertices (samples), edges (comparisons made) and 2-paths (dependencies created).

The clique graph. The standard tester is actually the full clique, comparing each possible pair of samples. In this dense graph we only need $\sqrt{|E|}$ vertices in order to have $|E|$ edges. However, many 2-paths (and dependencies) are created along the way as well, $\Theta\left(|E|^{3/2}\right)$. For this specific case, it is already known the two affects can be balanced to obtain optimal (asymptotic) sample complexity.

Disjoint cliques. Another interesting graph (used in some sense in [16]) is actually a union of disjoint cliques. This graph turns out to be quite useful. For once, it makes perfect sense in a simultaneous model, where each player process her own samples, sending a short summary to the referee. Surprisingly, it arises in other models as well.

A perfect matching. This graph relates to taking a fresh pair of samples each time we wish to make a new comparison, which leaves us with a set of completely independent collision indicators. Indeed, in this graph there are no 2-paths at all. Not only this tester minimizes the dependencies – it actually overdoes it. Doing so, it pays a price in sample complexity: the number of vertices we have is $2\,|E|$, much larger than the clique, for instance.

The star graph. With a fixed number of edges, this graph is actually the way to *maximize* the amount of 2-paths and dependencies – which makes it a very poor comparison graph. Indeed, the tester it induces is equivalent to drawing one element from P, and comparing it to many other samples, assessing the probability of this element. This test is indeed ill-advised against distributions were $3/4$ of the elements have mass $1/n$.

The full bipartite graph. Another graph that could be considered is the full bipartite graph, $G = (V_1 \sqcup V_2, V_1 \times V_2)$. Here again we have a free parameter (the size $|V_1|$, which determines $|V_2| = |E| / |V_1|$). It ranges from a star-graph (for $|V_1| = 1$) to a balanced graph (for $|V_1| = |V_2| = \sqrt{E}$), where the last one functions asymptotically similarly to the clique. It appears that Theorem 3 only gets optimal testers (minimizing the number of comparisons) from these graphs when they are balanced. To some extent, the test of [12] in the streaming model is based on such a graph. However, they use additional steps that seem crucial for the analysis. more details are given in 3.2, where it is also shown that one can test uniformity in the streaming model solely via collisions counting, using a whole different comparison graph.

1.2 Models and Results

Our main result is a method that produces well-performing uniformity testers in various models. In this paper we show a list of uniformity testers in different models, specified below. We also show limitations of our method for most of these models, which point towards a conclusion that no better comparison graphs could have been chosen (up to constant factors in the sample complexity of the induced tester). These limitations rely on a conjecture that no matter the shape of the graph, enough comparisons always must be made.

We emphasize that for any model in which a lower bound is known (for any method, not necessarily collision-based testing), the testers produced by our method are optimal, up to $\mathrm{poly}(1/\epsilon)$ factors. As far as we know, no testers in the literature are tight with the current lower bounds for these specific cases. Thus, it could be the case that collision-based testing achieves optimal results (even in terms of ϵ) for all the models we consider. A more thorough discussion is given in Section 4.2.

Equipped with the structural theorem, we consider various models and devise a uniformity tester in each one. The key idea here is to translate the constraints of each model into the comparisons we are able to perform, or differently put: a structural limitation on the comparison graph. Doing so will guide us how to choose a "good" comparison graph for this specific model. Having a structure in mind, two formalities are left: (i) Prove that calculation of Z, T (the number of collisions, and the threshold value) can be done in the model; (ii) Calculate our desired complexity measure (which changes from model to model), and optimize the parameters of the chosen graph (e.g., if the graph is a clique, determine the size of the clique).

Standard processors. In Section 3.1, we deal with the classic and the simultaneous model. First we use the classic model as a warm-up. Since this is done in [14], we add to the mix a small insight: our framework (which allows adjusting the threshold) actually provides slightly better constants when placing the threshold much lower than $1/2$ (at roughly $\tau = 1/9$).

For this model, it was shown by [13] that testing with high precision can be done faster than it would have using standard amplification. However, they use estimation of the empirical distance from uniform over the sample set. It is unclear whether collision-based testing is fit for this task. We do not pursue this direction here.[2]

[2] One reason is that our work focuses on the regime that uses new and different comparison graphs, other than the clique graph. This direction would probably involve analysis that is specific for the clique graph, where all samples can be compared with one another.

We then move to the simultaneous case where multiple players each send a short message to the referee, based on their own samples. The referee then needs to output a decision about the underlying distribution. In this model we want to find a good exchange for the number of players, the number of samples each player gets, and the length of the messages. For example, if the messages can be arbitrarily long, each player can send her entire sample set and the problem becomes trivial. For this reason, the two papers to first consider (independently) testing in this model, had a very different focus. In a preliminary version of [5, 6], only the case of a *single sample* per player was considered, and their algorithms indeed rely on different and interesting strategies, but not collision counting – as this strategy is irrelevant for this regime. In [16] a different approach was taken, where all messages were fixed to a *single bit*, but each player gets multiple samples (and in fact, their algorithm does rely on collisions in some sense, but it does not count them accurately, and does not fall under the umbrella of our definition for collision-based testers).

For our use, as oppose to both these view, we allow both parameters to be larger. We allow multiple samples per player, and show that using a short message (not a single bit, but not much longer), one can devise an efficient tester.

We also consider the asymmetric cost variant of this model, which aims to model the case where not all players have the same sampling capabilities (say, one player might gather samples much faster than another). As we measure our complexity by number of samples, we might as well think of the sampling process as being the bottleneck which we wish to hasten. When trying to minimize the sampling *time* instead, it is only natural to generalize the model to the case where some players are more efficient than others. Other than this motivation, a more technical motivations exists: a natural reduction from the LOCAL model. The LOCAL model is a standard interactive model. Uniformity testing in this model was first considered in [16], which gave a very natural reduction to a simultaneous model where players have different sampling rates.

We remark that many works in the simultaneous model consider communication trade-offs, when assigning only a *single* sample for each party. Our method does not currently extend to this framework, although one might consider integrating it with other methods. For example, one method (e.g., in [6]) uses random hash to a smaller domain. One might consider collisions on this domain instead of the original one. these meta-collisions can be counted within the communication constraints.

Some works in this regime (single sample) focus on privacy aspects of testing (e.g., [2, 7]). This line of research should be irrelevant for our method (and even the extension mentioned above), as any detection of a collision (even on a smaller domain) would immediately give away non-trivial information about the sample.

Memory-constrained processors. In Section 3.2, we deal with a different type of processors: *memory-constrained* processors. When observing a single processor of this type, we end up with the streaming model (as described in [12]). At least one scenario which motivates the simultaneous model is seemingly very coherent with such constrained processors. Thinking of a network of sensors, or remote devices, gathering samples – it is quite comprehensible that these processors are not only limited by their ability to communicate with the data center, but also by their ability to store the entire data observed between consecutive reports (in this scenario, the data center is the referee performing the test periodically to detect anomalies). For this reason, we also consider the case of a simultaneous model, where each processor has a small memory budget. In this new model, we easily devise again an efficient uniformity tester.

Testing in an interactive model. Lastly, in the full version of this paper, we use the structural theorem to show how on certain graphs one can solve uniformity in the CONGEST model faster than what was previously known to be possible. Specifically, if the communication network has k players and diameter D, and each players start with one sample, the best known algorithm runs in $O(D + n/(k\epsilon^4))$ rounds [16]. The improvement we suggest is an algorithm that takes $O(D)$ rounds, and works in specific networks that have a certain topological characteristics. Moreover, we show a simple detection procedure of $O(D)$ rounds, that can be used to recognize such a good topology. This means that *any* network can use the detection procedure first, and then either proceed as in [16], or switch to the faster ($O(D)$) algorithm whenever it is guaranteed to perform well.

1.3 Related Work

The task of uniformity testing, as well as the collision-based tester for it, were introduced in [9] (and implicitly in [21]). Later on, upper and lower bounds on the sample complexity of the problem were given by [24] (for most values of ϵ), showing that the optimal sample complexity is in fact $s = \Theta\left(\sqrt{n}/\epsilon^2\right)$, where n is the world size, and ϵ is a proximity parameter. A preliminary version of [26] giving a tester that achieves this complexity for any value of ϵ. The last two papers used two different testers: the first relied on the number of *distinct* elements in the sample set (this test is somewhat dual to counting collisions), and the second on a modified χ^2 tester. It was then shown by [14] that the collision-based tester does in fact achieve optimal sample complexity too.

In the past few years there has been a growing interest in distribution testing under various computational models, including collaborative testing in multiparty model, the streaming model, privacy aspects of testing, and others ([5, 6, 16, 8, 2, 22, 3, 4, 1, 23, 17, 7] and more). We mention in more details the works concerning uniformity testers in models we pursue.

In a preliminary version of [5, 6] and [16] each, independently, the task of uniformity testing was considered in a simultaneous communication model. In this model, all players receive samples from the same global distribution P, and all players report to a referee based on their own samples. The referee in turn uses the reports to output (with high probability) whether P holds some property.

In both lines of work, the focus was uniformity testing, but using two different perspectives: the former zeroes in on one sample per player, where the trade-off in question is between the number of bits each player is allowed in his report, and the number of players needed to the testing process. We think of the number of bits as too small to describe the sampled element fully. In this setting, a full description of two samples is never available to a single player (not even the referee). We do note that the tester given there relies on a looser notion of collisions (Taking a coarser division of $[n]$ into subsets).

In the later, the focus is different: one now fixes instead the communication to *one* bit per player. Now, the trade-off is between the number of players and the number of samples each one of them takes. The upper bound devised there discuss each player taking the right amount of samples, and notifying the referee 0 if no collision occurred, and 1 otherwise. In some sense, the referee ends up counting collisions (notice the count is trimmed, as one player might see more than 1 collision, but is only able to report 0 or 1). This tester is then used as a black-box to solve uniformity testing in the classic distributed models (CONGEST and LOCAL), in a setting where each player initially draws one sample.

One other paper to specifically discuss uniformity testing is [12], in which a streaming version of the problem is defined, as well as another distributed version, in a blackboard model, where all players are privy to the messages sent by others. As opposed to the simultaneous models mentioned above – here several rounds of communication are allowed. As it turns out, the streaming algorithm, as well as another algorithm for the distributed version, boil down yet again to counting collisions under the limitations of the model.

In addition to these, quite a few works had the focus of showing impossibility results both in the classic and the entire variety of models. For out interest, we mention [15, 24] for the classic version, as well as [22] for the simultaneous model (with multiple samples per machine), and [12, 3] for the streaming model.

2 Preliminaries

Throughout this text, we discuss uniformity testing using collision-based testers.

We let $[n] := \{1, 2, \dots, n\}$, and use $\Delta([n])$ to denote the set of distributions over the set $[n]$. As we only care about the support size (rather than the values), it is enough to consider $P \in \Delta([n])$ as possible input distributions. For a distribution P over $[n]$, we write P_i for the probability of the i^{th} element.

An (n, ϵ)-uniformity tester is an algorithm $\mathcal{A}$ that given oracle access to some unknown distribution $P \in \Delta([n])$, takes $q = q(n, \epsilon)$ samples from P and satisfy the following:

- If the input distribution is $P = U_n$, the uniform distribution over $[n]$, then $\mathcal{A}$ outputs YES with probability at least 3/4.
- If $\|P - U_n\| \geq \epsilon$, which means the distribution P is ϵ-far from uniform), then $\mathcal{A}$ outputs NO with probability at least 3/4.

The distance used here is L_1. Meaning, for two distribution P, Q, the distance is $\|P - Q\| = \sum_{i=1}^{n} |P_i - Q_i|$.

To formalize our notion of a *collision-based* tester, we take a fresh point of view of the sampling process. The key object in our analysis is the *comparison graph*, which is simply an undirected graph $G = (V, E)$, where we think about V as a set of placeholders for samples and E as the pairs of samples which are chosen to be compared with one another.

▶ **Definition 1** (Sampling process, collision indicators)**.** *Given a comparison graph $G = (V, E)$ and an input distribution $P \in \Delta([n])$, the sampling procedure S_P is described as a random labeling of the vertices according to P. We denote by $S_P : V \to [n]$, the process for which $\forall i. S_P(v_i) \sim P$, independently from one another. We end up with $S_P(v_1), \dots, S_P(v_{|V|})$ which is a set of $|V|$ i.i.d samples from P.*

Moreover, we define for any edge $e = (u, v)$ in E, the collision indicator $\mathbb{1}_e^P := \mathbb{1}_{S_P(u) = S_P(v)}$.

When P is clear from context, we simply write $S(u)$ for the sample sitting in vertex u, and $\mathbb{1}_e$ for the collision indicator.

We are now ready to give a formal definition for a collision-based tester, which relies on a set of comparisons (not necessarily between all pairs of samples), and compares Z - the amount of collisions, with some threshold value T.

▶ **Definition 2** (Collision-based tester)**.** *Fix n, ϵ. For any comparison graph $G = (V, E)$ and real number $0 \leq \tau \leq 1$, we define the algorithm $\mathcal{A} = (G, \tau)$ as follows: upon receiving as input $|V|$ i.i.d samples from P (given by $S_P(v_1), \dots, S_P(v_{|V|})$), it computes the following:*

$$Z := \sum_{e \in E} \mathbb{1}_e \ , \qquad\qquad T := |E| \cdot \left(\frac{1 + \tau\epsilon^2}{n} \right) \ ,$$

and outputs YES if $Z < T$, and NO otherwise.

The restriction $\tau \in [0,1]$ will help us deal with technicalities, but we note that it is rather intuitive. Indeed, as we will see later, only for these values the expectation of Z is lower than T for the good input (uniform distribution), and higher than T for *all* bad inputs.

Throughout, we will focus on properties of the graph and of our input distribution. We denote by $|E_G|$, $|V_G|$ the number of edges and vertices in G, and by $c(G)$ the number of times a 2-path appears as a subgraph in G. We count each 2-path twice, for its 2 automorphisms, and so we need to count "directed" 2-paths (so e_1, e_2 and e_2, e_1 are both counted). Formally, we can write $c(G) = \left|\left\{(u,v,w) \in \binom{V}{3} \mid (u,v),(v,w) \in E\right\}\right|$.

For the distribution P over $[n]$, with P_i for the probability of the i^{th} element, we denote the collision probability $\mu_P = \sum_{i=1}^{n} P_i^2$, and the three-way collision probability $\gamma_P = \sum_{i=1}^{n} P_i^3$. For brevity, whenever G or P are clear from context, we simply write $|V|$, $|E|$, μ, γ.

2.1 Models of Computation

In all our results we are concerned with distribution testing (and specifically uniformity testing), and we deal with various models. Therefore, in the following lines we specify in which way samples are taken in each model, and in what way the answer of the algorithm needs to be declared (where a good tester is the one that outputs YES (resp. NO) with high probability whenever the samples are taken from a YES (resp. NO) distribution).

The centralized model. The centralized model is the classic model. In this model one processor receives all samples $s_1, \ldots, s_q$ (in comparison graph notations, we think of $s_i = S(v_i)$, and $q = |V|$), and is tasked with outputting a proper answer according to the underlying input distribution. The complexity measure we wish to minimize in this model is q, the number of samples.

The simultaneous model. The second model we consider is the simultaneous model, where k players (processors) each draw individual samples unseen by all other players. Each player sends a short message to the referee. The referee then aggregates the messages and outputs the answer. Formally, each player is tasked with V_i where the whole set of vertices in G is $V = \sqcup_{i=1}^{k} V_i$. The samples of processor i are then $\{S(v)\}_{v \in V_i}$. Each processor can send a message a_i which is a function of its samples, and a referee receives all messages $a_1, \ldots, a_k$ and outputs the answer. The simultaneous first appeared in the context of testing independently in [16] and preliminary version of [5, 6], where in the first $|a_i| = 1$ meaning each player is allowed to send one bit, and in the latter $|V_i| = 1$ meaning each player gets exactly one sample. We take the same point of view as in [16], but we remove the restriction of 1 bit and allow a longer (but still short) message instead.

The number of players k is given as a parameter, and our goal is to minimize the number of samples *per player*, where all players get the same amount of samples: q/k (we think of it as sort of parallelization of the sampling process). We also consider the asymmetric-cost variant, where each player has an individual cost for each sample it draws (we think of this cost as the time it takes to draw each sample), and we wish to minimize the cost (or time) of the entire sampling process.

We also discuss the case of memory-constrained processors, also referred to as the streaming model, where this distribution testing was recently considered in [12].

Memory-constrained processor. In the memory-constrained model, each processor receives its samples as a stream, and once a sample is dealt with it is gone forever (this is the one-pass variant of the streaming model). A processor can only use a limited amount at each given moment, denote by m (and measured by memory bits). We think of $m' = \lfloor m/(2\log n)\rfloor$ as the number of samples we can store with half the memory (we leave the other half for other calculations). The complexity measure of this model is the number of samples needed to complete the testing task.

We also consider a simultaneous model where each processor is memory-constrained, receiving its samples as a stream, and using its m bits of memory it needs to come up with a message a_i to send to the referee once all samples are seen. The referee then receives all messages and outputs an answer. We stick to the case where all processors are of the same type and therefore have the same constraints of m bits.

3 Results

In this section we go over numerous applications of our method. For each model we go over the same phases: we start with intuition as to which comparison graph G is fit to this model, and we go on to show how one can simulate a collision-based algorithm $\mathcal{A} = (G, \tau)$. By simulating the algorithm we mean that by the end of the calculation, some processor will be able to compute both the number of collisions (Z) and the fitting threshold (T), so it is able to output the answer. The last part is optimizing parameters, where first order parameters are those of G, and in some application we also give focus to second order parameters (choosing τ).

The strength of the method comes from the following structural theorem that gives sufficient conditions for a comparison graph inducing a good uniformity tester:

▶ **Theorem 3.** *Fix a domain size n and a proximity parameter ϵ. If the following hold for an algorithm $\mathcal{A} := (G, \tau)$:*

1. $|E| \geq \frac{4n}{\tau^2 \cdot \epsilon^4}$,
2. $|E| \geq \frac{16n}{(1-\tau)^2 \cdot \epsilon^4}$, *and*
3. $\frac{c(G)}{|E|^2} \leq \frac{(1-\tau)^2 \epsilon^2}{16\sqrt{n}}$,

then $\mathcal{A}$ is an ϵ-uniformity tester.

We direct the reader to the full version for the proof. For the most part, it is a generalization of the one used in [14] to show the original collision-based tester is optimal (in our notations, the original tester over q samples is simply the algorithm $\mathcal{A} = (K_q, 1/2)$). While generalizing the proof we leave not one, but *three* separate conditions on a general comparison graph, that together guarantee it induces an (n, ϵ)-uniformity tester. This supplies a better, multi-dimensional understanding of how well a collision-based algorithm is guaranteed to perform. We note that if one is willing to ignore constants, one could simply fix $\tau = 1/2$ and merge the first two conditions into one. However, interestingly for our method other values of τ (usually smaller) guarantee slightly better constants. We leave all 3 conditions separate to maintain maximum flexibility when proving application of the theorem.

A specific comparison graph that is key to our algorithms is the one of *disjoint cliques*. Indeed, our strategy will be "perform any comparison you can" which sometimes simply means we have several bulks of samples, where in each bulk all pairs can be compared. To this end, we also give a more specific version of Theorem 3, for graphs that have this structure:

▶ **Lemma 4.** *Fix n, ϵ. Fix a comparison graph $G = \bigsqcup_{i=1}^{\ell} G_i$, where each G_i is isomorphic to K_q, for some $q \geq 3$. If the following hold for an algorithm $\mathcal{A} := (G, \tau)$:*

1. $q\sqrt{\ell} \geq \frac{\sqrt{12}\sqrt{n}}{\tau \cdot \epsilon^2}$
2. $q\sqrt{\ell} \geq \frac{\sqrt{48}\sqrt{n}}{(1-\tau) \cdot \epsilon^2}$
3. $q\ell \geq \frac{24\sqrt{n}}{(1-\tau)^2 \epsilon^2}$

then $\mathcal{A}$ is an ϵ-uniformity tester that uses $|V|$ samples.

Here again we leave the 3-conditions version in order to be able to adjust the threshold parameter for optimizations. However, here all 3 conditions are quite similar, pointing to the following simple corollary:

▶ **Corollary 5.** *Fix n, ϵ. Fix a comparison graph $G = \bigsqcup_{i=1}^{\ell} G_i$, where each G_i is isomorphic to K_q, for some $q \geq 3$. For each constant τ, the algorithm $\mathcal{A} := (G, \tau)$ is an ϵ-uniformity tester, if it holds that*

$$q\sqrt{\ell} \geq 35\sqrt{n}/\epsilon^2$$

The proofs of these are also omitted, and appear in the full version of the paper. We go on to show how these statements are used to devise testers in various models.

3.1 Standard Processors

3.1.1 Centralized Model

As a warm up, we re-prove the original collision-based tester works, in term of the comparison graph, and using 4. To add a small twist, we show that under our analysis, better sample complexity is guaranteed when using a biased threshold (meaning, taking $\tau \neq 1/2$). Let us denote $q := |V|$, which is the number of samples drawn, and our complexity measure for the model. The following is rather straightforward:

▶ **Corollary 6.** *One can test uniformity using $q = \Theta\left(\frac{\sqrt{n}}{\epsilon^2}\right)$.*

Proof. We simply use the lemma for $\ell = 1$ (one big clique), and we get 3 conditions of similar form, and in particular:

$$q \geq \max\left\{\frac{\sqrt{12}}{\tau}, \frac{\sqrt{48}}{1-\tau}, \frac{24}{(1-\tau)^2}\right\} \cdot \frac{\sqrt{n}}{\epsilon^2}$$

Which means that choosing e.g., $q = \frac{100\sqrt{n}}{\epsilon^2}$ with $\tau = 1/2$. We note that the tester can easily compute Z, T and therefore execute the collision-based algorithm (G, τ), for any value of τ.

An added perk here, is that one can optimize τ over the three conditions to reduce sample complexity by a constant factor. Even though the guaranteed constant is somewhat of an artifact of the proof, it is still interesting to see that $\tau = 1/9$ would reduce the constant from 100 to roughly 35 (while simple optimization over τ would reduce it even a bit more, for some irrational threshold value). ◀

3.1.2 Simultaneous Model

In the simultaneous model, k players are each given oracle access to the distribution P. After taking samples, each player is allowed to send a short message to a referee, who then needs to output the right classification for P (with high probability).

We give our focus to the variant posed in [16]: what is the number of *samples per player* needed to test uniformity? The "single collision" algorithm devised for that question only requires *one bit* from each player, and indeed it was shown to be optimal in [22]. However, this algorithm is somewhat delicate: first, the range of the parameter k is limited (it cannot be too high or too low, with regards to n, ϵ); second, if the number of players k is not accurately known to all players, the algorithm breaks. This is because unlike most results in the classic setting, the analysis here actually requires each player to use a specific amount of samples, but not more than that. Because the players can only communicate a single bit, everything else must be set in advance given the problem's parameters.

To that end, we relax the model, and allow each player to send more than a single bit, but still only a small number of bits is allowed. These would allow each player to send the number of collisions she saw (rather than *whether* a collision occurred). Our algorithm works for any parameter k, and can adjust to the case where each player is not exposed to the exact value of k, but rather to an approximation of it.

The model at hand imposes very concrete limitations on our comparison graph: one cannot compare a sample from one process to a sample of the another process. This means having k players solving the problem in a parallel way, is equivalent to having a comparison graph whose number of connected components is *at least* k. Followed by the intuition of "compare every pair you have", the graph we use is the disjoint cliques graph, with k disjoint cliques. Our measure of complexity is the number of samples each player used, q', which is translated to be the size of each one of our cliques.

▶ **Corollary 7.** *In the simultaneous model, one can test uniformity using* $q' = \Theta\left(\frac{\sqrt{n}}{\sqrt{k}\epsilon^2}\right)$ *samples per player, where each player is allowed to send* $\Theta(\log(1/\epsilon))$ *bits to the referee.*

Proof. We use Lemma 4, where each player has an independent sample set of size q' and compares all the pairs. We have k players in total doing so.

Our first goal is to show how to simulate a collision-based tester (G, τ) in this model. This will be possible whenever G is made of at least k vertex-disjoint connected components $G = \bigsqcup_{i\in[k]} G_i$.

Now, we can write $Z = \sum_i Z_i$ where $Z_i := \sum_{e\in E_i} \mathbb{1}_e$, and each Z_i is calculated by the *ith* player and then sent to the referee who simply sums them up to produce Z. Computing T is easy: the algorithm is known to all, and specifically the values $\tau, |E|$ are known to the referee. Now she simply needs to output the decision $\mathbb{1}_{Z<T}$.

Our next step is to quantify the communication and sampling cost of said algorithm. We aim at a communication cost of roughly $\log(1/\epsilon)$ bits per player. We note that Z_i can theoretically be very large, however if it surpasses T, then obviously $Z > T$, and the referee can reject. To this end we designate a specific string to say "too large" and instead each player sends Z_i only up to a value of T. This amount can be communicated using merely $\log(T)$ bits by each player.

We now calculate the sample complexity, measured in samples *per player*. We use Lemma 4 with q' and $\ell = k$, to know that it is enough if we satisfy:

1. $q' \geq \frac{\sqrt{12}\sqrt{n}}{\tau\cdot\sqrt{k}\epsilon^2}$
2. $q' \geq \frac{\sqrt{48}\sqrt{n}}{(1-\tau)\cdot\sqrt{k}\epsilon^2}$
3. $q' \geq \frac{24\sqrt{n}}{(1-\tau)^2 k\epsilon^2}$

Which is enough to show the asymptotic sample complexity we desire.[3]

[3] Assuming large enough k, the third condition pose asymptotically weaker requirement, and so optimizing τ on the first two would yield the optimal guarantee for $\tau = 1/3$, getting us to $q' = \left\lceil \frac{\sqrt{108}\sqrt{n}}{\sqrt{k}\epsilon^2} \right\rceil$ samples per player.

Since the third condition is looser than the first two, we actually choose the number of samples per player, q', such that $|E| = \Theta\left(n/\epsilon^4\right)$. Since $\tau\epsilon^2 \leq 1$, this means that our threshold is not too large

$$T = \Theta\left(\frac{n}{\epsilon^4}\right) \cdot \left(\frac{1+\tau\epsilon^2}{n}\right) = \Theta\left(1/\epsilon^4\right),$$

and so the simulation of the tester only requires each player to send $\log\left(\Theta(1/\epsilon^4)\right)$ bits to describe the number of collisions she saw. ◀

▶ Remark 8. As apparent from [5, 6, 22], when messages are r bits long, the correct value to look for in the sample complexity is 2^r (and sometimes $\sqrt{2^r}$). For this reason, when discussing communication, we explicitly write expressions such as $\log\left(\Theta(1/\epsilon^4)\right)$ instead of a more general $\Theta\left(\log(1/\epsilon)\right)$. This would prove useful when comparing our results to known lower bounds, as is done in Section 4.2.

3.1.3 Asymmetric-Cost Model

This variant is best described and motivated as follows: we think of the sampling process as time consuming, and each of the k players now has her own sampling rate, meaning that some players are able to draw samples faster than others. We describe the sampling rate vector $R = (R_1, \ldots, R_k)$, and we let each player draw her own number of samples $(s_1, \ldots, s_k)$. The complexity measure is the time dedicated for the sampling process, denoted t. Within t time, player i collects exactly $q_i := \lfloor R_i \cdot t \rfloor$ samples.

Simulating the tester (calculations of Z, T) works just as before. The big difference is that now each player has a different amount of samples (depending on her rate R_i) and thus a different clique size. We omit the calculations which can be found in the full version, and merely state the result we obtain:

▶ **Corollary 9.** *In the simultaneous model, one can test uniformity in time* $t = \Theta\left(\frac{\sqrt{n}}{\epsilon^2 \|R\|_2}\right)$, *where* $\|R\|_2 = \sqrt{R_1^2 + \cdots + R_k^2}$, *and each player sends* $\Theta\left(\log\left(1/\epsilon\right)\right)$ *bits to the referee.*

It is interesting that this tester actually generalizes the previous two. Indeed, taking rate vectors $R = (1, \ldots, 1)$ gives the symmetric model, and $R = (1, 0, \ldots, 0)$ the centralized one.

3.2 Memory-Constrained Processors

3.2.1 Centralized Model with Memory Constraints

In the streaming version of the problem, introduced in [12], the samples from the distribution P arrive in a stream, and our processor can only store m bits of memory. We wish to stream as few samples as possible and still output with high probability whether P is uniform or ϵ-far. For our purposes, we use $m' = \lfloor m/2(\log(n))\rfloor$, the number of *samples* that can be stored with half of the memory space (the other half will be used to perform calculations).

The restriction this model imposes is that when a certain sample is being processed, it can only be compared with the $O(m')$ samples that are currently stored. This intuition can be formalized, showing that any comparison graph used in the streaming model must have a bounded *average* degree (the maximal degree can be arbitrarily high: store the first sample and compare with all others). Indeed, we later quantify this statement (see Claim 16, in Section 4.1.2). For now, we just use it as intuition for the right comparison graph.

In [12] the full bipartite graph was used (with few modifications): they store a batch of samples V_1 and compare the rest of the stream (V_2) while comparing each new samples to all samples in V_1. For a partial range of m, they achieve sample complexity of $\Theta\left(n/(m'\epsilon^4)\right)$, which is shown to be optimal for an even more limited range parameter m. It was left as an open question whether this sample complexity can be attained for a wider range of values for m. For the upper bound, we answer this question in the positive. We use not only a different analysis, but rather a whole different comparison graph, one that also has a low average degree. For simpler representation, we again turn to the disjoint cliques graph used before. The induced algorithm is this: we allocate half the memory for samples and half for calculations. for the first half we take batches of m' samples, and make all comparisons in between them. We then delete them entirely and make room for a new batch. The number of comparisons is a compact information, that can be easily calculated and stored on the second half of our memory tape.

Once again we leave the full details in the full version of this paper and sum up the differences. The calculations are somewhat similar to before. The one major difference (that results with a different-looking result) is the fact we now express the sample complexity with respect to parameter m (the size of each clique), whereas before we used k (the number of cliques) as a parameter.

Since the first half of the tape must store samples, and the second half must count collisions all the way up to the threshold $T = \Theta(1/\epsilon^4)$, the result only applies for $m = \Omega\left(\log(1/\epsilon)\right)$. Whenever $m' \geq c\sqrt{n}/\epsilon^2$ there is not need (nor a possibility) to split our samples to batches, and instead we use one large batch to test uniformity such as in the centralized (standard) model. Combining all the above, we obtain the following result:

▶ **Corollary 10.** *In the streaming model with memory m , one can test uniformity using*

$$q = \Theta\left(\max\left\{\frac{n\cdot\log(n)}{m\epsilon^4}, \sqrt{n}/\epsilon^2\right\}\right)$$

samples, as long as we have $m = \Omega\left(\log(1/\epsilon)\right)$.

Note that our result has similar complexity as in [12], but it widens the range of acceptable values for m. They had the requirement of $m = \Omega\left(\log(n)/\epsilon^6\right)$, which is far larger than the humble requirement posed on our tester. We leave open an interesting question posed also in [12]: is it possible to test for uniformity in the scarce regime, and if so – what is the sample complexity required to do so? (The scarce regime is essentially $m = o\left(\log(n)\right)$. Though, to comply with our result - e.g., for absurdly small values of ϵ - it can be redefined to $m = o\left(\max\left\{\log(1/\epsilon), \log n\right\}\right)$)

3.2.2 Simultaneous Model with Memory Constraints

The main difficulty in this section seems to be handling the multiple parameters: on top of n, ϵ, we use k processors, each of which can store at most m bits of memory (or m' samples), and is allowed a short one-sided communication to the referee.

We state the result and omit the proof (which is given in the full version).

▶ **Corollary 11.** *In the memory-constrained simultaneous model, whenever $m = \Omega\left(\log(1/\epsilon)\right)$ bits of memory are allowed for each player, and $\log\left(1/\epsilon^4\right)$ bits of communication are allowed for each player, uniformity can be tested using*

$$\Theta\left(\frac{n}{k\epsilon^4}\cdot\max\left\{\frac{\log(n)}{m}, \frac{\sqrt{k}\epsilon^2}{\sqrt{n}}\right\}\right)$$

samples per player.

Note that in the regime of very small memory constraint, the problem suddenly "parallelize" perfectly: imagine taking $k \leq \Theta\left((n/\epsilon^4) \cdot (\log^2(n)/m^2)\right)$ machines with memory m per machine. In this case, each machine needs only a $1/k$ fraction of the amount of samples required for a single machine with memory m. This phenomena is explained by the observation that the real barrier (or scarce resource) in this regime is the overall storage, which indeed grows linearly with the number of machines used.

4 Limitations of the Method

In this section our goal is to better understand the possibilities (and impossibilities) of collision-based testing, in comparison to arbitrary testing methods (many of which were developed for specific testing task, as mentioned earlier in the text).

Here, we use Definition 2 to shift the discussion to graph terminology, focusing on the comparison graph. We leverage simple graph properties to show some limitations that apply when generating testers, such as done in Section 3.

The results hereinafter apply to testers where correctness is proven via our structural theorem. However, we next formulate a simple conjecture (regarding the necessity of making enough comparisons) that implies these results are true for any tester that answers Definition 2. In matter of fact, we will formally show that under the framework of Theorem 3, the comparison graphs we have chosen are essentially the best to answer the requirements. If the conjecture is proven to be true, we get the stronger results that this graphs are indeed optimal with respect to any tester from Definition 2).

In the discussion at the end of this section, we compare these results with known lower bounds (for arbitrary testers) under the same constraints. This section aims to both show optimality (or near-optimality) of the testers we develop, and more importantly: give a better understanding of collision-based testing and how strong it is, compared to an arbitrary tester.

We first go on to show some basic inequalities that hold in any simple graph (and in our comparison graphs as well). These will serve us for the rest of this section. The goal is to establish the inherent connection between the 3 sizes: $|V|, |E|, c(G)$ in any simple graph G.

4.1 Conditional Impossibility Results

We start by stating the following easy lemma that connects our quantities of interest in any simple graph.

▶ **Lemma 12.** *For any simple graph G, it holds that:*

1. $|E| \leq \frac{|V|^2}{2}$
2. $|V| \geq \frac{4|E|^2}{2|E|+c(G)}$
3. *if* $|V| \leq |E|$ *then* $c(G) \geq 2|E|$
4. *Whenever* $|V| \leq |E|$, *we have* $|V| \cdot c(G) \geq 2|E|^2$

We omit the proof here (it appears in the full version), and proceed to put these properties into use.

In order to understand the scope of possible applications of Theorem 3, we stick to graph notations, and combine the 3 different types of assertions concerning our comparison graph:

1. The requirements needed to apply Theorem 3.
2. The basic graph properties of Lemma 12.
3. Individual assertions that apply for the model at hand.

We note that all the results below only use the first requirement of Theorem 3, which asks for the total number of comparisons to be large enough. We believe this requirement to be inherent for *any* collision-based tester (rather than an artifact of our analysis). We formulate this belief as the following conjecture, which would strengthen the impossibility results of this section to be independent of the analysis (instead of limitation of Theorem 3, we get a lower bound for all collision-based testers as defined in Definition 2).

▶ **Conjecture 13.** *Any collision-based tester as in Definition 2 that tests uniformity with error at most* $1/4$ *must have* $|E| = \Omega\left(\frac{n}{\epsilon^4}\right)$

Note that it holds in two extremes: for maximum amount of dependencies (clique graph) it follows from any lower bound in the classic version of the task. On the other side, the perfect matching has no dependencies at all, and it induces a tester that draws a new pair of samples each time. As the collision probability is either $1/n$ or larger than $(1+\epsilon^2)/n.$ each comparison is like an independent coin toss with this biased (as observed by [6]). It is known for this problem that at least $\Theta\left(n/\epsilon^4\right)$ tosses are needed to decide the bias, which shows correctness of the conjecture for this comparison graph.

4.1.1 Standard Processors

Centralized model. As a warm-up, we turn to the classic model, where no constraints are involved. We easily see that Conjecture 13 helps. Using the first bullet in Lemma 12 gives the known lower bound in the classic version of the problem: $q = |V| \geq \sqrt{2\,|E|} = \Omega(\sqrt{n}/\epsilon 2)$.

Simultaneous model. In the simultaneous case, the main restriction is that of each player has her own comparison graph, as no comparisons can be made between two different players. This leads us to the following:

▶ **Corollary 14.** *Assuming Conjecture 13 holds, the number of samples per player of any collision-based uniformity tester in the simultaneous model is* $q' = \Omega\left(\frac{\sqrt{n}}{\sqrt{k}\cdot\epsilon^2}\right)$.

Proof. We recall the sample complexity here is the maximum amount of samples one player draws. Formally, we write the comparison graph as the union of k disjoint parts: $G = \bigsqcup_{i\in[k]} G_i$, where $G_i = (V_i, E_i)$, and so the complexity measure is simply $q' = max_{i\in[k]}\,|V_i|$.

We note, however, that $|E_i| \leq |V_i|^2/2$ for each component G_i, and therefore:

$$|E| = \sum_{i\in[k]} |E_i| \leq \sum_{i\in[k]} |V_i|^2/2 \leq k \cdot max_{i\in[k]}\,|V_i|^2/2 = k\cdot q'^2/2.$$

Plugging in Conjecture 13, we get

$$q' \geq \sqrt{\frac{2\,|E|}{k}} = \Omega\left(\frac{\sqrt{n}}{\sqrt{k}\cdot\epsilon^2}\right),$$

which ends the proof. ◀

Asymmetric cost model. The more elaborate version of the asymmetric-cost model also impose similar limitations, where the difference comes from the generalized definition of the complexity measure.

▶ **Corollary 15.** *We observe the asymmetric-cost simultaneous model, with sampling rate vector* $(R_1,\ldots,R_k)$. *If Conjecture13 holds, then any collision-based uniformity tester in this model must use sampling time of* $t = \Omega\left(\frac{\sqrt{n}}{\epsilon^2\|R\|_2}\right)$.

Proof. We again use $G = \bigsqcup_{i\in[k]} G_i$, where $G_i = (V_i, E_i)$. However, the complexity measure is the time t in which player i with rate R_i can obtain $|V_i| = q_i = t \cdot R_i$ samples.

Again, using the trivial edges-vertices inequality over each component G_i, we get

$$\forall i.\ |E_i| \le |V_i|^2/2 = t^2 \cdot R_i^2/2$$

and summing all together, we get

$$|E| = \sum_{i\in[k]} |E_i| \le \sum_{i\in[k]} t^2 \cdot R_i^2/2 \le t^2 \|R\|_2^2/2.$$

Joined with Conjecture 13, it concludes the proof

$$t \ge \sqrt{\frac{2|E|}{\|R\|_2^2}} = \Omega\left(\frac{\sqrt{n}}{\epsilon^2 \|R\|_2}\right).$$ ◀

4.1.2 Memory-Constrained Processors

Here we use a slightly more sophisticated argument, to show that the memory constraint can also be translated to graph notation. We emphasize that our desire is to show limitations of our framework, and so we relax the model and assume that comparisons are made on designated memory cells, in which we can only store m' element names. In order to count collisions *accurately*, we cannot expect to compress this data further. For example, $m' = o(\sqrt{n})$ and samples are drawn from the uniform distribution $P = U_n$, with 99% we need to write in our memory m' different elements, and this information cannot be compressed.

We go on to show how the memory constraint translates well:

▷ Claim 16. Let us assume a constrained machine can only store m' elements at a time, and it is able to accurately count collisions on a comparison graph $G = (V, E)$. Then it must be the case that $|E| \le m' \cdot |V|$.

Proof. w.l.o.g let us name the vertices, or samples, by their order in the stream $V = \{v_1, v_2, \ldots, v_s\}$, and w.l.o.g let us think of the edges in E as ordered pairs (We only write $(i.j) \in E$ for pairs where $i < j$).

Now, we note that at time t, upon processing the sample v_t, the memory can only store m' samples from the set $s(v_1), \ldots, S(v_{t-1})$. This means that the number of edges in E of the form (v_i, v_t) is at most m'. Now, we can count our (ordered) edges using the second item:

$$|E| = \sum_{v_j\in V} |\{(u, v_j) \in E \mid u \in V\}| \le \sum_{v_t\in V} m' = |V| \cdot m'$$

which completes the proof. ◁

Centralized model with memory constraints. We next apply this claim to give similar lower bounds in the following models.

▶ **Corollary 17.** *Assume Conjecture 13 holds. Any collision-based uniformity tester that can only store at most* $m' = \Theta(m/\log(n))$ *samples at any given time, must take at least* $q \ge \Omega\left(\frac{n\log(n)}{m\epsilon^4}\right)$ *samples.*

Proof. As our measure complexity is the total amount of samples, $q = |V|$, we can combine the claim above with Conjecture 13 to get:

$$q = |V| \ge \frac{|E|}{m'} = \Omega\left(\frac{n\log(n)}{m\epsilon^4}\right),$$

which ends the proof. ◀

Simultaneous model with memory constraints. We observe the combined model of simultaneous model with memory-constrained machines. We again restrict ourselves to a specific framework: each player uses her own graph $G_i = (V_i, E_i)$, where memory is allocated for sampled elements, and then all players send a short message to the referee. The entire comparison graph the algorithm is based on is $G = \bigsqcup_i G_i$.

▶ **Corollary 18.** *Any collision-based uniformity tester in a simultaneous model of k machines that can store up to m' samples each, must use*

$$q' = \Omega\left(\frac{n}{k\epsilon^4} \cdot \max\left\{\frac{\log(n)}{m}, \frac{\sqrt{k}\epsilon^2}{\sqrt{n}}\right\}\right)$$

samples per player, assuming Conjecture 13 holds .

Proof. We start be re-writing the desired expression:

$$q' = \Omega\left(\max\left\{\frac{\sqrt{n}}{\sqrt{k}\epsilon^2}, \frac{n\log(n)}{mk\epsilon^4}\right\}\right)$$

We show the two lower bounds separately. Indeed, the first lower bound can be derived directly from Corollary 14:

$$q' = \Omega\left(\sqrt{n/(k\epsilon^2)}\right)$$

For the second lower bound, we extend Corollary 17 instead. As each G_i is done by a machine with memory constraints, we apply Claim 16 to player i and get $|V_i| \geq |E_i|/m'$. We recall that our measure complexity is in fact $q' := \max_i |V_i|$, and as the maximum is greater than the average, we get:

$$q' \geq \frac{\sum_i |V_i|}{k} \geq \frac{\sum_i |E_i|}{m' \cdot k} = \frac{|E|}{m' \cdot k}$$

And plugging in Conjecture 13 on the entire graph $G = (V, E)$, we get

$$q' = \Omega\left(\frac{n}{m' \cdot k\epsilon^4}\right) = \Omega\left(\frac{n\log(n)}{m \cdot k\epsilon^4}\right),$$

concluding the proof. ◀

4.2 Discussion

We point out to the fact that in all 4 models (as well as the classic model), the limitation of this method coincide with the upper bounds we obtained in the previous section. This does not come as a surprise, as choosing the "right" comparison graph is easily made once the constraints of the model are understood. The rule of thumb is simply to compare all pairs that can be compared.

It is more interesting, though, to compare the collection of possible testers with the sub-collection of collision-based testers.We next brief through known impossibility results (for an arbitrary tester. It turns out for the most part, collision-based testers compete well with others. For the classical model, as already established in [13], collision-based testing is in fact optimal. For the two simultaneous models (with no memory constraints), the collision-based testers perform optimally in terms of n, k (or $n, (R_1, \ldots, R_k)$ in the asymmetric case). To the best of our knowledge, the only testers for these models to consider *multiple* samples per processor are the ones of [16]. For both models, the two papers show similar sample complexity, but there are two non-trivial differences. On one hand, the new testers use $\log\left(\Theta(1/\epsilon^4)\right)$ bits per communication, instead of a single one used in [16]. On the other hand, the new testers work for the full range of the parameter k (the number of players), whereas the previous results excluded extreme values: e.g., in the symmetric case it only works for $c/\epsilon^4 \leq k \leq c'n\epsilon^4$.

Lower bounds for both models are shown in [22], even for the case of r-bit messages. While the tester of [16] is an optimal one-bit protocol, ours is not known to be optimal r-bit protocol. This is true as the aforementioned lower bound weakens by a factor 2^r for the longer r-bit messages. In our case, we have $2^r = \Theta(1/\epsilon^4)$, which means there is a gap of ϵ^4 between the general lower bound, and the optimal collision-based tester we obtain. Despite the gap, no better tester is known for this amount of bits and $q > 1$ samples per player.

In the streaming model our tester attain the same sample complexity as the best known tester (that of [12]), but for a wider range for the parameter m. A matching lower bound can also be found in [12], but only for a more restricted range of the parameter m, whereas for the general case they give a weaker lower bound, which leaves an ϵ^2 gap between the general case and the optimal collision-based tester.

To the best of our knowledge, the simultaneous model with memory constraints was never considered in the context of distribution testing, and therefore there are no prior results. However, we conjure that similarly to before, the collision-based tester achieves optimal sample complexity in parameters n, k, m, with a possible poly(ϵ) gap that would pop, as before, due to the use of longer messages.

References

1 Jayadev Acharya, Sourbh Bhadane, Piotr Indyk, and Ziteng Sun. Estimating entropy of distributions in constant space. In *Advances in Neural Information Processing Systems*, 2019.

2 Jayadev Acharya, Clément Canonne, Cody Freitag, and Himanshu Tyagi. Test without trust: Optimal locally private distribution testing. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2067–2076, 2019.

3 Jayadev Acharya, Clément L Canonne, Yuhan Liu, Ziteng Sun, and Himanshu Tyagi. Interactive inference under information constraints. *arXiv preprint*, 2020. `arXiv:2007.10976`.

4 Jayadev Acharya, Clément L Canonne, and Himanshu Tyagi. Distributed signal detection under communication constraints. In *Conference on Learning Theory*. PMLR, 2020.

5 Jayadev Acharya, Clément L Canonney, and Himanshu Tyagiz. Inference under information constraints i: Lower bounds from chi-square contraction. *IEEE Transactions on Information Theory*, 2020.

6 Jayadev Acharya, Clément L Canonney, and Himanshu Tyagiz. Inference under information constraints ii: Communication constraints and shared randomness. *IEEE Transactions on Information Theory*, 2020.

7 Kareem Amin, Matthew Joseph, and Jieming Mao. Pan-private uniformity testing. In *Conference on Learning Theory*, pages 183–218. PMLR, 2020.

8 Alexandr Andoni, Tal Malkin, and Negev Shekel Nosatzki. Two party distribution testing: Communication and security. In *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

9 Tugkan Batu, Lance Fortnow, Ronitt Rubinfeld, Warren D. Smith, and Patrick White. Testing that distributions are close. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 259–269, 2000.

10 Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of computer and system sciences*, 47(3):549–595, 1993.

11 Clément L. Canonne. A survey on distribution testing: Your data is big. but is it blue? *Electronic Colloquium on Computational Complexity (ECCC)*, 22:63, 2015.

12 Ilias Diakonikolas, Themis Gouleakis, Daniel M Kane, and Sankeerth Rao. Communication and memory efficient testing of discrete distributions. *arXiv preprint*, 2019. `arXiv:1906.04709`.

13 Ilias Diakonikolas, Themis Gouleakis, John Peebles, and Eric Price. Sample-optimal identity testing with high probability. *CoRR*, abs/1708.02728, 2017. `arXiv:1708.02728`.

14 Ilias Diakonikolas, Themis Gouleakis, John Peebles, and Eric Price. Collision-based testers are optimal for uniformity and closeness. *Chicago Journal OF Theoretical Computer Science*, 1:1–21, 2019.

15 Ilias Diakonikolas and Daniel M. Kane. A new approach for testing properties of discrete distributions. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, 2016.

16 Orr Fischer, Uri Meir, and Rotem Oshman. Distributed uniformity testing. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, PODC '18. ACM, 2018.

17 Sumegha Garg, Pravesh K. Kothari, and Ran Raz. Time-Space Tradeoffs for Distinguishing Distributions and Applications to Security of Goldreich's PRG. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020)*, pages 21:1–21:18, 2020.

18 Oded Goldreich. *Introduction to Property Testing.* Cambridge University Press, 2017.

19 Oded Goldreich. The uniform distribution is complete with respect to testing identity to a fixed distribution. In *Computational Complexity and Property Testing.* Springer, 2020.

20 Oded Goldreich, Shari Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *Journal of the ACM (JACM)*, 45(4):653–750, 1998.

21 Oded Goldreich and Dana Ron. On testing expansion in bounded-degree graphs. *Electronic Colloquium on Computational Complexity (ECCC)*, 7(20), 2000.

22 Uri Meir, Dor Minzer, and Rotem Oshman. Can distributed uniformity testing be local? In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 228–237. ACM, 2019.

23 Varun Narayanan, Manoj Mishra, and Vinod M Prabhakaran. Private two-terminal hypothesis testing. *arXiv preprint*, 2020. `arXiv:2005.05961`.

24 L. Paninski. A coincidence-based test for uniformity given very sparsely sampled discrete data. *IEEE Transactions on Information Theory*, 54(10):4750–4755, 2008.

25 Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, 1996.

26 Gregory Valiant and Paul Valiant. An automatic inequality prover and instance optimal identity testing. *SIAM Journal on Computing*, 46(1):429–455, 2017.