Doubly Sub-Linear Interactive Proofs of Proximity

Noga Amir
Weizmann Institute of Science, Rehovot, Israel

Oded Goldreich
Weizmann Institute of Science, Rehovot, Israel

Guy N. Rothblum
Apple, Cupertino, CA, USA

Abstract

We initiate a study of *doubly-efficient* interactive proofs of *proximity*, while focusing on properties that can be tested within query-complexity that is significantly sub-linear, and seeking interactive proofs of proximity in which

1. The query-complexity of verification is significantly smaller than the query-complexity of testing.
2. The query-complexity of the honest prover strategy is not much larger than the query-complexity of testing.

We call such proof systems doubly-sublinear IPPs (dsIPPs).

We present a few doubly-sublinear IPPs. A salient feature of these IPPs is that the honest prover does not employ an optimal strategy (i.e. a strategy that maximizes the verifier's acceptance probability). In particular, the honest prover in our IPP for sets recognizable by constant-width read-once oblivious branching programs uses a distance-approximator for such sets.

2012 ACM Subject Classification Theory of computation → Interactive proof systems

Keywords and phrases Interactive Proof Systems, Interactive Proofs of Proximity, Query Complexity, Read Once Branching Programs, Sub-linear

Digital Object Identifier 10.4230/LIPIcs.ITCS.2025.6

Related Version *Full Version*: https://eccc.weizmann.ac.il/report/2024/143/

1 Introduction

This work combines the mind-frames of interactive proofs of *proximity* (i.e., IPPs) and *doubly-efficient* interactive proofs (de-IPs), while giving the notion of doubly-efficient a new meaning that focuses on the *query-complexity* of the honest prover strategy. Specifically, we focus on properties that can be tested by reading a small portion of the input (equiv., have query-complexity that is significantly sub-linear), and seek interactive proofs of proximity in which

1. The query-complexity of verification is significantly *smaller than* the query-complexity of testing.
2. The query-complexity of the honest prover strategy is *not much larger than* the query-complexity of testing.

In addition, we may seek analogous relations for the time-complexities; yet, we recall that, in the context of property testing, time-complexity is secondary to query-complexity (see [4, Sec. 1.3.1]).

A salient feature of (almost all) the IPPs that we present is that the honest prover does not employ an optimal strategy, because it cannot afford to read the entire input (per the second query-complexity condition). In particular, while an optimal prover strategy for the verifiers that we present achieves perfect completeness, our honest provers don't.

1.1 Our focus: query-efficient generation of proofs of proximity

Focusing on properties that can be tested in sub-linear query-complexity, we initiate a study of interactive proofs of proximity in which verification requires less queries than testing whereas proving does not require much more queries than testing. We recall the wider context first.

Property testing. The focus of property testing[6, 22] is on approximate decision procedures that read small portion of their input (see [4] for an introduction to the subject). For a predetermined property (i.e., a set of valid objects), approximate decision means distinguishing between objects that have the property and objects that are far from any object having this property.[1] Such procedures, called testers, are probabilistic and obtain local views of the object by performing queries; that is, the object is seen as a function and the testers get oracle access to this function (and thus may be expected to read only part of the object).

The insistence on procedures that read a small portion of the input reflects envisioning applications in which the input is huge and fetching it entirely is infeasible. In some of these applications, one may afford computation time that is almost linear in the size of the input, although fetching the entire input is still deemed infeasible.

Proofs. A generic question is whether proofs can offer verification that is more efficient than the corresponding decision. In the context of property testing, this refers to interactive proofs of proximity (IPPs)[3, 21]. In such proof systems, *verification should require significantly less queries than testing*, and suitable notions of completeness and soundness should hold.[2] Specifically, *completeness* means that inputs that have the property should be accepted with high probability (when the prover uses an adequate strategy), whereas *soundness* means that inputs that are far from the property should be rejected with high probability (no matter what strategy is employed by the prover). Indeed, in the completeness condition we consider a honest prover, whereas in the soundness condition we consider a cheating one.

Doubly-efficient proofs. Seeking to utilize proof systems in reality prohibits the use of arbitrary honest provers. Such applications require that the honest prover strategy be relatively efficient, a requirement that is captured by the term *double-efficiency*, which refers to the complexities of both the verifier and the honest prover. Given our postulate that it is infeasible to fetch the entire input and our focus on query complexity, this means that the honest prover strategy should also have query complexity that is significantly smaller than linear. We call such proof systems doubly-sublinear IPPs (dsIPPs).

Note that dsIPPs may exist only for properties that can be tested using small query complexity. This is the case because the interaction between the verifier and the honest prover (along with the queries they make) can be emulated by the tester. Hence, we focus on properties that can be tested using small query complexity and ask *for which of these properties we can obtain dsIPPs*?

---

[1] Distances are measured according to the relative Hamming measure; that is, $x \in \{0,1\}^n$ is considered ε-far from $y \in \{0,1\}^n$ if x and y differ on more than $\varepsilon \cdot n$ locations.

[2] In addition, one requires low communication complexity. In particular, this does not allow the prover to send the entire input to the verifier (a possibility that is not relevant in the context of double-efficiency that we consider here).

We stress that the notion of double-efficiency employed here is different from the notion employed in the context of IPs (see[10, 19, 5]) and also in studies of IPPs that made reference to doubly-efficient IPPs (e.g.[20]). In these prior cases, the reference was to polynomial running time of the honest prover, whereas we focus on its query complexity and on the case that it is small (and in particular is sub-linear).

1.2 Our Main Results

We construct doubly-sublinear IPPs (dsIPPs) for several problems. Specifically, for any property that can be decided by a constant-width read-once oblivious branching program (ROOBP), for approximating the input's Hamming weight, for telling whether a given function is a permutation, and for a relaxation of the graph bipartiteness problem in the bounded-degree model.

As mentioned earlier, the honest prover strategies in (almost all) the dsIPPs that we present do not employ an optimal strategy, because they cannot afford to read the entire input. In particular, while optimal prover strategies for these IPPs achieves perfect completeness (i.e., the verifier always accepts inputs that have the property), our honest provers do not achieve perfect completeness.

1.2.1 Protocol for Read-Once Oblivious Branching Programs

We construct a dsIPP for any property that can be decided by constant-width read-once oblivious branching programs (ROOBPs). Recall that in such branching programs, in each layer, all vertices are labelled by the same input variable (which is the one being read), and each input variable labels the vertices of at most one layer. A branching program decides a property Π if it accepts all inputs in the property and rejects all inputs that are not in the property. See Section 4.1 for a fuller discussion on ROOBPs.

Newman [16] showed that any property that can be decided by a constant-width ROOBP has a tester that makes $\mathsf{poly}(1/\varepsilon)$ queries, where the exponent of the polynomial is linear in the width of the branching program. We construct a dsIPP for any such property, where the verifier's query complexity is only $O(1/\varepsilon)$, which is optimal.[3]

▶ **Theorem 1** (dsIPP for ROOBPs, informal). *Let Π be a property that can be decided by a constant-width ROOBP, and let n and ε denote the input length and the proximity parameter. Then, for every $r : \mathbb{N} \to \mathbb{N}$, there exists an $r(n)$-round dsIPP for Π such that the verifier has query complexity $O(1/\varepsilon)$, the honest prover has query complexity $\widetilde{O}(n^{1/r(n)}) \cdot \mathsf{poly}(r(n)/\varepsilon)$, and the communication complexity is $O(n^{1/r(n)} \cdot r(n) \cdot (\log n)/\varepsilon)$.*

See Corollary 20 for a full and formal statement. Evidently, our protocol allows for a trade-off between the number of rounds (on one hand) and the query complexity of the prover and the communication (on the other hand). In particular, a logarithmic number of rounds suffices for getting the prover's query complexity and the communication to depend only poly-logarithmically on n. The verifier's runtime can be linear in the communication complexity assuming access to a suitable procedure specifying the structure of the ROOBP (see below).[4] The prover's runtime depends on the computational complexity of tolerant testing for properties decided by ROOBPs, see below.

[3] Consider the set property $\{0^i 1^{n-i} : i \in [n]\}$, the input $x = 0^{n/2}1^{n/2}$ and a honest prover strategy that refers to the input x but is also invoked as a cheating strategy on a random input r that is at Hamming distance 2ε from x. Then, an $o(1/\varepsilon)$-query verifier cannot distringuish the two cases.

[4] We remark that for a general non-uniform ROOBP no verifier can have a sublinear running time: for example, one layer in the ROOBP might make it go into a state that rejects all inputs. Without reading the entire ROOBP, the verifier has no hope of detecting whether there is a "universal rejection" layer.

Our protocol builds on an IPP of Goldreich, Gur and Rothblum[7], in which the honest prover reads the entire input. On the other hand, the IPP of[7] works also for unbounded width (but its communication complexity grows logarithmically with the width). We note that our dsIPP is actually *tolerant* (see below), whereas this was not the case for the protocol of [7] (or the tester of [16], see below).

A reduction to tolerant testing. An interesting feature of our IPP is that the honest prover strategy is reduced to *tolerant* testing (or distance approximation) for a property that can be decided by an ROOBP of (at most) the same width and length. A *tolerant* tester [17] for a property Π gets proximity parameters $\varepsilon_c < \varepsilon_f$ and is supposed to accept (w.h.p.) if the input is ε_c-close to the property, and to reject (w.h.p.) if the input is ε_f-far from the property. The complexity is usually measured as a function of the gap $\varepsilon_f - \varepsilon_c$ (and possibly also of the input length).

Theorem 1 follows by plugging any tolerant tester for constant-width ROOBPs into our general reduction (which is stated in Theorem 2). While, as noted above, Newman's tester for ROOBPs[16] is not tolerant, we show that that it can be made tolerant. This contribution of the current work is presented in the full version of the article.

Our reduction of the prover's strategy to tolerant testing ROOBPs actually gives a *tolerant* dsIPP of ROOBPs, where *Tolerant IPPs* are defined analogously to tolerant testers.[5] Fixing the ROOBP under consideration, the reduction assumes that the honest prover has access to a distance approximator that gets an ROOBP of (at most) the specified width and length, and approximates the distance of an input from the corresponding property (i.e., the property decided by the ROOBP). We need the deviation of the distance approximation to be $O((\varepsilon_f - \varepsilon_c)/r)$, and its error probability to be sufficiently small. The approximator only needs to operate on "sub-ROOBPs" of the original ROOBP.

Before stating the reduction, we also specify the access the verifier and the prover need to the ROOBP in order to get sublinear *runtimes*. We assume that the prover and the verifier have (unit cost) access to two procedures that *specify the structure of the ROOBP* as follows. The first procedure, given two nodes u and v in the ROOBP, returns a bit indicating whether there is a path from u to v. The second procedure gets as input the index i of a layer and returns the vertices in that layer. (We emphasize that these procedures do not depend on the input to the ROOBP: they ignore edge labels and refer to the ROOBP as a directed graph.)

▶ **Theorem 2** (Reducing tolerant dsIPP to distance approximation for ROOBPs, informal)**.** *For any constant w, let Π_w be a property that can be decided by a width w ROOBP, and let $\varepsilon_c, \varepsilon_f$ denote the proximity parameters. Suppose that there exists a distance approximator for w-width ROOBP (and its subgraphs) with query-complexity $Q_w(\delta, \eta, n)$ and time-complexity $T_w(\delta, \eta, n)$, when δ denotes the approximation parameter, η denotes the error probability bound, and the ROOBP has length n. Then, for every $r : \mathbb{N} \to \mathbb{N}$, there exists an $r(n)$-round tolerant dsIPP for Π_w such that*

- *The verifier's query-complexity is $O(1/\delta)$, and the honest prover's query-complexity is $O(n^{1/r(n)} \cdot Q_w(\delta, \eta)/\delta)$, where $\delta = (\varepsilon_f - \varepsilon_c)/3r(n)$ and $\eta = \delta/O(n^{1/r(n)})$.*
- *The communication complexity is $O(n^{1/r(n)} \cdot r(n) \cdot (\log n)/\delta)$.*

[5] Specifically, if the input is ε_c-close to the property, then the honest prover should convince the verifier to accept (w.h.p.); but if the input is ε_f-far from the property, then the verifier should reject (w.h.p.) regardless of the prover's strategy. We stress that, while in the context of testing, tolerant testing and distance approximation are equivalent (under suitable parameters), this is not the case for dsIPPs (see Section 2 and Remark 8).

- *Assuming that the verifier and the prover have access to procedures specifying the structure of the ROOBP (see above), the verifier's runtime equals the communication complexity (up to constant factors), and the honest prover's runtime is* $O(n^{1/r(n)} \cdot r(n) \cdot \mathsf{T}_w(\delta, \eta, n)) \cdot \widetilde{O}(1/\delta)$.

See Theorem 19 for a full and formal statement. As discussed above, Theorem 1 is obtained from the above reduction by plugging in a distance approximator for constant-width ROOBPs, which we construct based on Newman's (non-tolerant) property tester. While the reduction of Theorem 2 only has polynomial dependence on the width of the ROOBP, Newman's tester and our tolerant version of it have an exponential dependence on the width of the ROOBP (which is the reason that Theorem 1 holds only for constant width).[6] See the details in Section 4.6.

1.2.2 Protocol for Hamming Weight

We construct a dsIPP for the Hamming weight (HW) problem, where the input is a string $x \in \{0,1\}^n$ and a claim about its Hamming weight. Note that Hamming weight cannot be computed by constant-width ROOBP [18], so the results of Section 1.2.1 do not give a dsIPP for HW.

Approximating the (relative) Hamming weight up to distance ε requires $O(1/\varepsilon^2)$ queries in the standalone setting. We construct a dsIPP with $O(1/\varepsilon)$ verifier queries (which is optimal).

▶ **Theorem 3** (dsIPP for Hamming weight, informal)**.** *Let n and ε denote the input length and the proximity parameter for the Hamming weight problem (HW). Then, for every $r : \mathbb{N} \to \mathbb{N}$, there exists an $r(n)$-round dsIPP for HW such that the verifier has query complexity $O(1/\varepsilon)$ and the honest prover's query complexity and its runtime are both $\widetilde{O}(n^{1/r(n)} \cdot r^3(n)/\varepsilon^3)$. Furthermore, the verifier's runtime and the communication complexity are both $O(n^{1/r(n)} \cdot r(n) \cdot \widetilde{O}(1/\varepsilon))$.*

See Theorem 10 for a full and formal statement. This protocol also obtains a trade-off between the number of rounds and the honest prover's runtime. Taking the number of rounds to be logarithmic in n gives a *honest prover with query complexity and runtime that are poly-logarithmic in n*. The honest prover's query complexity and runtime grow cubically with (the reciprocal of) the proximity parameter. We wonder whether this dependence can be improved to quadratic (which would be optimal [1]).

Rothblum, Vadhan and Wigderson [21] also constructed an IPP for Hamming weight, but did not have a sub-linear honest prover. Our HW protocol borrows ideas from the IPP for ROOBP of[7]. Indeed, one can get an IPP for HW directly from their ROOBP protocol, by applying it to the $O(n)$-width ROOBP that computes HW. However, their protocol does not have a sub-linear prover.

1.2.3 Protocol for PERM

For $n \in \mathbb{N}$, the set PERM consists of all permutations over $[n]$. Here we consider the problem of testing (resp., verifying) whether a function $f : [n] \to [n]$ is in PERM. Note that PERM has a tester of complexity $O(\sqrt{n/\varepsilon})$, which we outline below, whereas a query lower bound of $\Omega(\sqrt{n})$ follows from[13].[7] Furthermore, PERM has two different (1-round) IPPs, which were presented in[13] and[7], respectively. We review them next:

[6] In contrast, Theorem 2 holds also for varying width, but in that case the complexities have a multiplicative factor of poly(w).

[7] See also a direct proof in[23, Apdx A].

1. In the IPP of[13] the verifier selects uniformly at random a point $r \in [n]$, and sends r to the prover, who is supposed to return its f-preimage; the verifier accepts if and only if the prover's answer is mapped by f to r.
2. In the IPP of[7], the verifier selects uniformly at random a point $r \in [n]$, queries f on it, and sends $y \leftarrow f(r)$ to the prover, who is supposed to return its f-preimage; the verifier accepts if and only if the prover's answer equals r.
   (This IPP, unlike the previous one, utilizes prover-oblivious queries.)

In both cases, $f \in$ PERM is always accepted, whereas functions that are ε-far from PERM are rejected with probability $\Omega(\varepsilon)$. (In both cases, the protocol is repeated $O(1/\varepsilon)$ times to yield an IPP.)

More importantly, in both cases, the honest prover finds the required f-preimage by querying f on all points (or practically so). Our initial conjecture was that an IPP for PERM cannot have a verifier that uses $o(\sqrt{n})$ queries and an honest prover that makes $o(n)$ queries. Interestingly, this "working conjecture" is wrong.

▶ **Theorem 4** (dsIPP for PERM)**.** *For every $\alpha \in (0, 0.5)$, there exists a 1-round IPP for PERM that has a verifier that uses $O(n/\varepsilon)^{0.5-\alpha}$ queries and an honest prover that uses $O(n/\varepsilon)^{0.5+\alpha}$ queries.*

The communication and time complexity of both parties is $\widetilde{O}((n/\varepsilon)^{0.5+\alpha})$.

Sketch of the proof of Theorem 4. The straightforward tester for PERM consists of selecting $q = O(\sqrt{n/\varepsilon})$ random points in $[n]$, querying the function $f : [n] \to [n]$ on them, and rejecting if and only if collisions are found (among the f-images).

Evidently, any $f \in$ PERM is accepted with probability 1, whereas (as can be seen later) any f that is ε-far from PERM is rejected with high constant probability. A similar analysis for the soundness case implies that for any f that is ε-far from PERM, if we select $O(n/\varepsilon)^{0.5+\alpha})$ random points, then we expect to see $\Omega(n^{2\alpha})$ collisions and that these collisions involve $\Omega(n^{2\alpha})$ disjoint pairs of points. This leads us to the following protocol.

1. The verifier selects $p = O(n/\varepsilon)^{0.5+\alpha}$ (distinct) random points in $[n]$, denoted $s_1, ..., s_p$, and sends them to the prover,
2. The prover queries the input $f : [n] \to [n]$ on these p sample points, and sends the answers $(a_1, ..., a_p) \leftarrow (f(s_1), ..., f(s_p))$ to the verifier.
3. The verifier rejects if it sees a collision (i.e., if $a_i = a_j$ for some $i \neq j$).
4. Otherwise, it sub-samples $m = O(n/\varepsilon)^{0.5-\alpha}$ of the original points, queries f on each of these m samples, and accept if and only if all answers match the prover's answers (i.e., if $f(s_i) = a_i$ for the i's it sub-sampled).

Clearly, $f \in$ PERM is always accepted. Turning to the soundness analysis, we fix an arbitrary function f that is ε-far from PERM and let $C \stackrel{\text{def}}{=} \{x \in [n] : |f^{-1}(f(x))| > 1\}$ denote the set of points that form collisions under f. Then, $|C| > \varepsilon \cdot n$; actually, $|C| - |f(C)| > \varepsilon \cdot n$, because modifying f to a permutation requires changing its value on all but a single point in $f^{-1}(y)$ for every $y \in f(C)$.

An analogous consideration applies to the sample of p points, denoted $S = \{s_1, ..., s_p\}$, that is sent to the prover. Specifically, the answers provided by the prover must disagree with the values of f on at least $|S| - |f(S)|$ points, because for each $s \in S$ the prover must provide different answers to all the other points in S that are in $f^{-1}(f(s))$ (since otherwise the verifier rejects in Step 2). Towards upper-bounding $|f(S)|$, we observe that it is upper-bounded by $|f'(S)|$, where $f' : [n] \to [n]$ is defined such that for every $x, x' \in [n]$ it

holds that (i) $f'(x) \neq f'(x')$ whenever $f(x) \neq f(x')$ and (ii) for each x that forms a collision under f, the restriction of f' to $f^{-1}(f(x))$ is 2-to-1 except on at most one element (in case $|f^{-1}(f(x))|$ is odd). Now, we lower-bound $|S| - |f'(S)|$ by the number of collisions of S under f' (i.e., $|\{\{i,j\} \in \binom{[p]}{2} : f'(s_i) = f'(s_j)\}|$)

First note that the number of collisions of $[n]$ under f' is at least $\sum_{y \in f(C)} \lfloor |f^{-1}(y)|/2 \rfloor$, which is at least $|C|/3 > \varepsilon n/3$. Hence, with high probability over the choice of $S \in \binom{[n]}{p}$, we get $\Omega(\varepsilon \cdot p^2/n) = \Omega((n/\varepsilon)^{2\alpha})$ collisions of S under f'. In this case, there are $\Omega((n/\varepsilon)^{2\alpha})$ disjoint pairs of points in S such that the elements in each pair have the same f'-image (and hence also the same f-image).

Wishing to avoid rejection in Step 2, the prover must cheat on the values of the $\Omega((n/\varepsilon)^{2\alpha})$ foregoing points (in S), but (with high probability) at least one of these points will appear in the sub-sample that the verifier selects in Step 4 (since the sub-sample rate is $m/p = O(\varepsilon/n)^{2\alpha}$). Hence, in Step 4, when quering the function f on this sub-sample, the verifier detects this cheating and rejects (w.h.p.). This completes the proof of Theorem 4.

1.2.4 Protocol for Bipartiteness

We construct a dsIPP for a *relaxation* of graph bipartiteness in the bounded degree model. In this model (see, e.g.,[4, Sec. 9.1]), the input is an undirected n-vertex graph of constant maximum degree d. The input graph is represented by its incidence function $g : [n] \times [d] \to [n] \cup \{0\}$ such that $g(v,i)$ is the i-th neighbor of the vertex v (or 0 if v has less than than i neighbors). The prover and the verifier have query access to this function. The distance between two n-vertex graphs is the ratio (over $dn/2$) of the number of edges on whose presence or absence they disagree.

The promise problem we consider is that the input graphs are rapidly-mixing graphs; that is, graphs in which a random lazy walk of logarithmic length starting at any vertex reaches any other vertex with probability $\Theta(1/n)$. (An ℓ-step lazy walk is described by a ℓ-long sequence over $[2d]$ such that $i \in [2d]$ indicates a step that moves from the current vertex v to its i^{th} neighbor (in case v has at least i neighbors) and staying in place if v has less than i neighbors.)

▶ **Theorem 5** (dsIPP for bipartiteness). *Let n and ε denote the number of vertices and the proximity parameter. Then, there exists a 1-round dsIPP for bipartiteness on rapidly-mixing graphs in the bounded-degree graph model in which the verifier's query complexity is $O(\log(n)/\varepsilon)$, and the honest prover's query and time complexity are $\widetilde{O}(\sqrt{n}/\varepsilon)$. Furthermore, the verifier's runtime is polylog$(n)/\varepsilon$, and the communication complexity is $O(\log(n)/\varepsilon)$.*

Related work. Goldreich and Ron [9] showed a $\Omega(\sqrt{n})$ lower bound on the query complexity of testing bipartiteness.[8] A later work of the same authors[8] shows a tester with $\widetilde{O}(\sqrt{n}) \cdot \mathsf{poly}(1/\varepsilon)$ query complexity. We remark that their tester works for general graphs, without needing the rapidly-mixing condition. Rothblum, Vadhan and Wigderson [21] constructed an IPP for an intermediate relaxation of bipartiteness, where the YES case includes all the bipartite graphs, but the NO case includes only the graphs that are both far from bipartite and rapidly-mixing. In their IPP, the verifier's query complexity is $O(\log(n)/\varepsilon)$, but the

[8] Their lower bound, which is stated for general 3-regular graphs, also holds under the additional promise that the graphs are rapidly-mixing (see the exposition in [4, Sec. 9.3.1]). In particular, the distribution of NO cases is concentrated on expander graphs (see [4, Clm. 9.18.1]). A similar analysis shows that the YES cases are bipartite expanders (w.h.p.)

honest prover is not sub-linear: it has to read the entire graph. Our proof of Theorem 5 uses the [21] protocol, but shows a sublinear-time implementation for the honest prover, where the implementation works for graphs that are both *rapidly-mixing* and bipartite.

Proof of Theorem 5: The sublinear-time honest prover. We first recall the[21] protocol. In the basic test, the verifier picks a random vertex s, and performs a lazy random walk of length $\ell = O(\log(n))$, which is long enough to guarantee the "rapidly-mixing" condition (see above), reaching a vertex t. The verifier sends s and t to the prover, asks the prover to recover the parity of a (simple) path that leads from s to t, and accepts if and only if the answer equals the parity of the number of "real moves" in the lazy walk (i.e., moves in which the walk did not stay in the current vertex). If the graph is bipartite, then the prover, who can read the entire graph, can always answer correctly by checking whether s and t are on the same side of the bipartition. As shown in[21], if the graph is rapidly mixing and ε-far from bipartite, then a cheating prover will fail with probability $\Omega(\varepsilon)$. In order to reduce the soundness error (from $1 - \Omega(\varepsilon)$ to $1/3$), this basic test is repeated $r = O(1/\varepsilon)$ times in parallel (and the verifier accepts if and only if it accepted in all invocations of the basic test).

We construct a sub-linear time honest prover for the basic test. Upon receiving the vertices s and t, the prover performs $w = O(\sqrt{n} \cdot \log(1/\varepsilon))$ independent random walks of length ℓ starting at s, as well as w independent random walks of length ℓ starting at t. Let W_s (resp., W_t) denote the set of vertices traversed in (the union of all) the walks that started at s (resp., t). The prover checks if there is a vertex v in the intersection of W_s and W_t (i.e., a vertex encounted in a walk that started at s as well as in a walk that started at t). If so, then the prover has found a walk from s to t (going through v), and it replies with the parity of the number of real moves on this walk.

Soundness follows directly from the soundness of the [21] protocol (since we did not change the verifier). We now argue for completeness (alas not perfect completeness). If the graph is bipartite, then all paths from s to t have the same parity. Thus, whenever the prover finds a path from s to t (i.e., when the intersection is non-empty), then it returns the correct parity. It remains to show that in each of execution of the basic test (i.e., for each s and t sent by the verifier), the intersection of W_s and W_t is non-empty with probability at least $1 - (1/3r)$. We show this in Claim 6, which is where we use the condition that the graph is rapidly-mixing, and it follows that the verifier accepts in all r repetitions with probability at least $2/3$.

▷ Claim 6. If the graph is rapidly-mixing, then, for every two vertices $s, t \in [n]$, the probability, over the honest prover's random walks, that $W_s \cap W_t = \emptyset$ is at most $1/3r$.

Proof. We focus on the sets $W'_s \subset W_s$ and $W'_t \subset W_t$ of *terminal* vertices reached by the random walks (i.e., the final vertex in each walk), and show that these subsets intersect with probability at least $1 - (1/3r)$. Using the rapidly-mixing condition, observe first that taking $w = O(\sqrt{n} \cdot \log(1/\varepsilon)) = O(\sqrt{n} \log r)$ independent walks from s, with probability at least $1 - (1/6r)$, the number of distinct vertices reached (i.e., the size of the set W'_s) is $\Omega(\sqrt{n})$. Assuming that this event occurs, and recalling that each of the random walks starting at t is rapidly-mixing, we infer that, with probability at least $\Omega(1/\sqrt{n})$, such a walk terminates in a vertex that resides in W'_s. Recalling that there are $O(\sqrt{n} \cdot \log r)$ walks starting at t, the probability that they all land outside of W'_s is thus at most $1/6r$. The claim follows. ◁

1.3 Further Related Work

Goldwasser, Rothblum, Shafer and Yehudayoff [11] studied interactive proofs for approximate verification of the results of a machine-learning computation on an unknown underlying distribution. The verifier and the honest prover both have sampling access to the unknown distribution (in some of their results the honest prover also has membership queries, see also [12]). While their model is quite different from ours, there is a similarity in spirit: if we think of the unknown distribution as a "huge input" to the proof system, then they are also concerned with proof systems where verification is more efficient than the computation being verified (a learning problem), and proving is not much more expensive than performing the same computation. In particular, proving is sub-linear in the size of the "input" (the unknown distribution). The settings, however, are quite different: both in the access to the input (querying an unknown function vs. sampling from an unknown distribution), and in the types of tasks studied (approximate decision vs. machine learning). We stress that other recent works on verifying properties of unknown distributions [2, 14, 15] do *not* study *honest provers with sublinear sample complexity.*

1.4 Technical Overview

In this section provide overviews of our dsIPPs for Hamming Weight and for properties that are decidable by ROOBPs. Recall that the dsIPPs for PERM and for a relaxed version of Bipartiutness were sketched in prior sections.

We start by outlining our IPPs for Hamming Weight (HW), and then extend its underlying ideas to obtain IPPs for ROOBPs. Both IPPs follow an abstract idea that underlies the IPPs of[21, 7], but deviate from it in a significant manner. Loosely speaking, the IPPs of[21, 7] rely on partitioning the original property to sub-properties that refer to parts of the original input. However, while the IPPs of[21, 7] call for an honest prover strategy that finds optimal partitions of the property to sub-properties, we shall use partitions that are sub-optimal, because these sub-optimal partitions can be found more efficiently (by the honest prover).

IPPs for Hamming Weight (HW). Our r-round IPPs proceed by recursion, where for a parameter k (to be set to $n^{1/r}$), the current input x is partitioned to k equal-length blocks, denoted $x_1, ..., x_k$. Clearly, the Hamming weight of x, denoted $\mathtt{wt}(x)$, equals the sum of the Hamming weights of the x_i's (i.e., $\sum_{i\in[k]} \mathtt{wt}(x_i)$). Hence, wishing to prove that $\mathtt{wt}(x)$ equals w, an optimal prover determines the corresponding weights of the x_i's, denoted $w_1, ..., w_k$, sends the w_i's to the verifier, who selects a random i, and the parties proceed to prove that the weight of x_i equals w_i. (Indeed, after r iterations, the verifier can check the claim, which refers to a single bit, by making a single query.) Needless to say, determining these w_i's requires reading the entire input x, which we want to avoid. Instead, we consider a query-efficient honest prover that obtains approximations of the desired w_i's (or obtains the exact value when $|x| = k$, which happens after $r-1$ recursion steps). This requires relaxing the verification procedure so that it does not reject in case the sum $\sum_i w_i$ does not equal w (but is rather only close to it).

As in[21], the soundness analysis relies on the fact that if $|\mathtt{wt}(x) - w| > \Delta$, then, for every sequence of w_i's such that $\sum_{i\in[k]} w_i = w$, it holds that $\sum_{i\in[k]} |\mathtt{wt}(x_i) - w_i| > \Delta$; equivalently, if x is ε-far from having weight w, then, for every sequence of w_i's such that $\sum_{i\in[k]} w_i = w$, on the average, x_i is ε-far from having weight w_i. Note, however, that if we allow $\sum_{i\in[k]} w_i$ to deviate from w by say $\varepsilon' \cdot |x|$, then, on the average, x_i is $(\varepsilon - \varepsilon')$-far from having weight w_i. Hence, at the bottom of the recursion, the expected distance of single bits from their claimed weight is $\varepsilon - r \cdot \varepsilon'$, which means that the verifier rejects with such probability. Using $\varepsilon' = \varepsilon/2r$ and repeating the protocol $O(1/\varepsilon)$ times, we obtain the desired IPP.

IPPs for constant-width ROOBP. As in the case of HW, we wish to proceed by recursion. Indeed, in the r-round IPP of[7], on current input x, an optimal prover determines the path in the current ROOBP that accepts x, which (as before) is partitioned to equal-length strings $x_1, ..., x_k$. This path defines k sub-ROOBPs that each accept the corresponding x_i's. Alas, determining these sub-ROOBPs requires reading the entire input x, which we want to avoid.

Using the fact that the ROOBP has bounded width, it follows that, for each $i \in [k]$, there is a bounded number of sub-ROOBPs (i.e., the square of the width bound) such that x_i is accepted by (at least) one of them. Using a distance approximator for (constant-width) ROOBPs, the honest prover can find, for each $i \in [k]$, a sub-ROOBP such that x_i is close to being accepted by this sub-ROOBP. This means that the foregoing description has to be modified: In subsequent iteration of the recursion it does not necessarily hold that the current x is accepted by the current ROOBP; it is only the case that the current x is close to being accepted by the current ROOBP. But in this case, for $x = (x_1, ..., x_k)$, it does not necessarily hold that each x_i is close to being accepted the corresponding sub-ROOBPs, but it is rather the case that their average distance from these sub-ROOBPs is small; that is, if x is ε-close to the current ROOBP, then for some $\varepsilon_1, ..., \varepsilon_i$ such that $\sum_{i\in[k]} \varepsilon_i/k = \varepsilon$, each x_i is ε_i-close to a corresponding sub-ROOBP, but it is not necessarily the case that $\varepsilon_i \approx \varepsilon$ for each $i \in [k]$.

Recall that the honest prover does not find these ε_i's, but rather finds their approximate values (as well as the corresponding sub-ROOBPs). Specifically, for width bound b, the honest prover considers an auxilary graph G with $k+1$ layers, index $0, 1, ..., k$, such that the i^{th} layer of G contains the vertices that are at the $i \cdot k$ layer of the current ROOBP. Each pair of vertices in adjacent layers of G represents a possible sub-ROOBP, where pairs in layers $i-1$ and i of G represent a sub-ROOBP that reads x_i. For each such pair, the honest prover estimates the distance of x_i from being accepted by the corresponding sub-ROOBP, where these estimates are obtained by invoking the distance-approximator (for ROOBPs). Using a shortest path algorithm, the honest prover finds sequence of intermediate vertices $(v_1, ..., v_{k-1})$ in G such that the average distance of the x_i's from the corresponding sub-ROOBPs, denoted $(B_1, ..., B_k)$, is minimal, and sends $(v_1, ..., v_{k-1})$ to the verifier along with the corresponding distances. (As in the case of HW, the verifier will select $i \in [k]$ uniformly at random, and the parties will proceed to prove the corresponding claim (i.e., that x_i is ε_i-close to being accepted by the corresponding sub-ROOBP).)

2 Preliminaries and Definitions

For strings $x, y \in \{0,1\}^n$, the relative Hamming distance between x and y is the fraction of coordinates in which they disagree (we often refer to this as the *distance* for short). If this distance is at most ε, then we say that x is ε-close to y, and otherwise we say that x is ε-far from y. We define the distance of x from a (non-empty) set $S \subseteq \{0,1\}^n$ as its distance from the closest y in S, and we define the string being ε-close and ε-far from the set analogously. We extend these definitions from strings to functions by identifying a function with its truth table.

A property Π (or a language) is a set of strings of varying lengths (i.e. in $\{0,1\}^*$). The approximate decision problem for Π is deciding, for specified proximity parameters, whether an input string is close or far from the property. See [4] for an introduction to property testing: the field that studies algorithms of sublinear query complexity for such approximate decision problems.

▶ **Definition 7** (Interactive Proof of Proximity (IPP, tolerant)). *An IPP is a protocol between two probabilistic parties, a prover P and a verifier V who both get an input length n and a joint input $x \in \{0,1\}^n$. The verifier has query access to x (and explicit access to the input length n). The parties interact and at the end of the interaction the verifier accepts or rejects. The protocol is a (tolerant) IPP for a property Π and for proximity parameters $\varepsilon_c, \varepsilon_f : \mathbb{N} \to \mathbb{R}$ if for every input length n and every input $x \in \{0,1\}^n$:*

- ***Completeness:*** *If x is $\varepsilon_c(n)$-close to the property Π (in relative Hamming distance), then the verifier, after interacting with the prover P, accepts with probability at least $2/3$ (the probability is over the prover's and the verifier's coin tosses). If the verifier, interacting with the honest prover, accepts every input in the language with probability 1 then we say that the IPP has* perfect completeness.
  In this work we focus on IPPs where the honest prover only has query access to the input x (similarly to the verifier, it gets the input length n as an explicit input).
- ***Soundness:*** *If x is $\varepsilon_f(n)$-far from the property Π, then for every cheating strategy P^*, the verifier V, after interacting with the prover P^*, rejects with probability at least $2/3$ (the probability is over the verifier's coin tosses, w.l.o.g. the cheating prover can be taken to be deterministic, and can have explicit access to the entire input).*

A non-tolerant IPP is one where $\varepsilon_c = 0$. In this case, we refer to a single proximity parameter $\varepsilon = \varepsilon_f$ (the parameter ε_c is implicit). Testers are viewed as a special case of IPPs in which there is no prover (or no interaction with it).

The protocol's complexity measures include the (honest) prover's and the verifier's query complexities and runtimes, the communication complexity and the round complexity (the number of back-and-forth communication rounds). These complexities are typically measured as a function of the gap $(\varepsilon_f(n) - \varepsilon_c(n))$ between the proximity parameters and of the input length.

Our focus in this work is on protocols where the query complexities of the verifier and the honest prover are as small as possible. In particular, we focus on properties that have testers of sublinear query complexity and require that the verifier's query complexity should be smaller than the tester's. The prover's query complexity should be as close as possible to the tester's. We refer to these as *doubly sub-linear* IPPs (dsIPPs, see the discussion in the Introduction).

▶ Remark 8. We emphasize that, while in the context of testing, tolerant testing and distance approximation are equivalent [17] (under suitable parameters), this is not the case for IPPs. The reason for this divergence is that a tolerant IPP only provides a one-sided guarantee: it can affirmatively convince the verifier of *upper bounds* on the input's distance from the property, but it does not convince the verifier of a *lower bound* on the distance.

▶ Remark 9. We remark that in the study of standard IPPs (where the honest prover can read the entire input), if we allow linear communication complexity, then the query complexity can always be tiny. This is because the prover can send the entire input to the verifier. The verifier receives this alleged input and verifies that it is close to the real input by checking consistency for a few random locations (these are the only queries made to the real input). If the alleged input is close to the real input, then the verifier can simply check if the alleged input is in the property (or, for tolerant protocols, that it is at the appropriate distance from the property). For dsIPPs it is not clear that linear communication always allows for a sublinear verifier. In particular, the aforementioned strategy cannot be utilized because the honest prover doesn't know the entire input and cannot send it. Thus, dsIPPs with linear (or even polynomial) communication may be an interesting object for further study.

3 dsIPP for the Hamming Weight Problem

In this section, we present a doubly sub-linear IPP (dsIPP) for the Hamming Weight Problem. Specifically, given a claimed weight $\sigma \in \mathbb{N}$, a proximity parameter $\varepsilon > 0$ and an oracle access to an input $x \in \{0,1\}^n$, we present an r-round IPP, with $r = \log(n)$ and with poly-logarithmic communication complexity, in which V verifies that $\mathtt{wt}(x) = \sigma$ using query complexity of $O(\frac{1}{\varepsilon})$, the honest prover P uses poly-logarithmic query complexity. We note that we use r to represent the number of (pairs) of message exchanges (and not the total number of messages sent).

First, in Section 3.1, we present a standard IPP for the Hamming Weight Problem. This IPP (which is not doubly sub-linear) uses a divide-and-conquer approach, and in it the prover P computes exact weights for sub-sequences of the input x. Then, in Section 3.2, we present a warm-up towards a dsIPP: We amend the standard IPP by allowing P to approximate the Hamming weight of the sub-sequences, and relaxing the accepting conditions of V in a way that doesn't compromise the completeness and soundness of the protocol. Unfortunately, in this IPP the prover P still has query complexity of $\Omega(n)$. Lastly, in Section 3.3, we present the actual dsIPP, which applies the IPP presented in Section 3.2 recursively, thus improving the query complexity of P without compromising the query complexity of V.

3.1 The Standard IPP

Given some σ, we want to verify that $\mathtt{wt}(x) = \sigma$. First, we observe that when we partition x to k consecutive equally-length parts, $x_1, ..., x_k$, the following holds:

- On the one hand, if $\mathtt{wt}(x) = \sigma$, then $\sum_i \mathtt{wt}(x_i) = \sigma$.
- On the other hand, if $|\mathtt{wt}(x) - \sigma| > \varepsilon n$, then for any $(\sigma_1, ..., \sigma_k)$ such that $\sum_i \sigma_i = \sigma$, it holds that $\sum_i |\mathtt{wt}(x_i) - \sigma_i| > \varepsilon n$.

Given the above, we consider the following IPP: P sends $(\sigma_1, ..., \sigma_k) \leftarrow (\mathtt{wt}(x_1), ..., \mathtt{wt}(x_k))$ to V. Then, V verifies that $\sum_i \sigma_i = \sigma$, and if so, selects at random $i \in [k]$, and verifies that $\mathtt{wt}(x_i) = \sigma_i$ (by reading the entire x_i). Note that if $\mathtt{wt}(x) = \sigma$ and V interacts with P, then for every i, it holds that $\mathtt{wt}(x_i) = \sigma_i$, and the verifier always accepts. However, if $|\mathtt{wt}(x) - \sigma| > \varepsilon n$ and $\sum_i \sigma_i = \sigma$, it holds that $\sum_i \frac{\mathtt{wt}(x_i) - \sigma_i}{k} > \varepsilon \cdot \frac{n}{k}$. Then, the average deviation of the claimed weight of x from the actual weight of x translates to a deviation on a corresponding fraction of random i's, hence guarantees that the verifier rejects with probability at least ε. To increase the rejection probability, we can repeat this procedure for $O(\frac{1}{\varepsilon})$ times.

The query complexity of the verifier is the length of a single x_i times the number of repetitions, which gives $O(\frac{n}{\varepsilon \cdot k})$. However, to compute $(\mathtt{wt}(x_1), ..., \mathtt{wt}(x_k))$, the prover must access to the entire input x, which yields query complexity of n.

3.2 A Warm-Up towards a dsIPP

In this section, we aim to achieve poly-logarithmic query complexity for the honest prover P without compromising the query complexity of the verifier V: We amend the interaction described in Section 3.1 by allowing P to *approximate* the Hamming weight of the sub-parts of x. However, at the end of this section, we explain why P still has query complexity of $\Omega(n)$. In Section 3.3, we show how to solve this issue and present the actual dsIPP.

First, we observe that for a deviation parameter $\delta > 0$ and an error probability parameter $\eta > 0$, there exists a (δ, η)-approximator for the Hamming weight of $x \in \{0,1\}^n$ with query complexity $O(\frac{\log(\frac{1}{\eta})}{\delta^2})$ and error probability η: Given the input x and the deviation parameter

δ, the approximator chooses, uniformly at random, $O(\frac{\log(\frac{1}{\eta})}{\delta^2})$ indices of x, and outputs the (normalized) number of indices i for which $x[i] = 1$. Using Chernoff bound, we can show that with probability at least $1 - \eta$, the output of the approximator deviates from $\mathtt{wt}(x)$ by at most δn.

Based on this approximator, we amend the interaction such that the prover provides *approximated* weights to the verifier, by using the approximator with some predefined error probability $\eta > 0$ and deviation parameter $\delta > 0$ (we fix both parameters later). That is, P sends $(\sigma_1, ..., \sigma_k)$ to V, such that for every $i \in [k]$, σ_i is a (δ, η)-approximation for $\mathtt{wt}(x_i)$. We also change the verifier's accepting conditions in the interaction, so that V allows this deviation (and doesn't reject P's approximations). That is, V now verifies that $|\sigma - \sum_i \sigma_i| \leq \delta n$, chooses $i \in [k]$ uniformly at random, and verifies that $|\mathtt{wt}(x_i) - \sigma_i| \leq \frac{\delta \cdot n}{k}$ (by reading the entire x_i). Similar to the standard IPP, we repeat the protocol for $O(\frac{1}{\varepsilon})$ times.

3.2.1 Correctness

The completeness of the protocol still holds (but with a completeness error): If $\mathtt{wt}(x) = \sigma$ and we interact with P, then for a single invocation of the protocol the following holds: For every $i \in [k]$, with probability at least $1 - k \cdot \eta$, it holds that $|\sigma_i - \mathtt{wt}(x_i)| \leq \frac{\delta \cdot n}{k}$, and therefore also $\sum_i |\mathtt{wt}(x_i) - \sigma_i| \leq \delta n$, and the verifier accepts. Since we invoke the protocol for $O(\frac{1}{\varepsilon})$ times, then the verifier accepts in all invocations with probability at least $1 - t \cdot \eta$ where $t = O(\frac{k}{\varepsilon})$. Thus, we can set $\eta \leq 0.01 \cdot \frac{1}{t} = O(\frac{\varepsilon}{k})$, and the verifier accepts with high constant probability in all invocations.

To show the soundness of the protocol also holds, we observe that if $|\mathtt{wt}(x) - \sigma| > \varepsilon n$, then for any $(\sigma_1, ..., \sigma_k)$ such that $|\sigma - \sum_i \sigma_i| \leq \delta n$, it holds that $\sum_i |\mathtt{wt}(x_i) - \sigma_i| > (\varepsilon - \delta)n$. Then, the average deviation of the claimed weight of x from the actual weight of x, beyond the allowed deviation, translates to a deviation on a corresponding fraction of random i's. Since the verifier rejects if $|\mathtt{wt}(x_i) - \sigma_i| > \frac{\delta n}{k}$, we can set $\delta = \frac{\varepsilon}{3} < \frac{\varepsilon}{2}$ and infer that the verifier still rejects with probability at least $\Omega(\varepsilon)$ in a single invocation. Since we invoke the protocol for $O(\frac{1}{\varepsilon})$ times, we get that the verifier rejects with high constant probability in at least one of the invocations.

3.2.2 Query Complexity

Now, let us analyze the query complexity of V and P:

- The query complexity of V is $O(\frac{n}{\varepsilon k})$, because we read a sub-sequence of the input of length $\frac{n}{k}$ for $O(\frac{1}{\varepsilon})$ repetitions.
- The query complexity of P is $O(\frac{k \cdot \log(\frac{k}{\varepsilon})}{\varepsilon \cdot \delta^2})$, since we invoke the approximator with error probability parameter $\eta = O(\frac{\varepsilon}{k})$ for k times in each of the $O(\frac{1}{\varepsilon})$ repetitions.

Since we can $(\delta, 0.1)$-approximate the Hamming weight of x without any interaction using query complexity $O(\frac{1}{\delta^2})$, we can perform ε-test for the Hamming weight of x using query complexity $O(\frac{1}{\varepsilon^2})$. Therefore, for the IPP to be meaningful, we want the query complexity of V to be $o(\frac{1}{\varepsilon^2})$. To achieve this in our setting, we must require $k = \omega(\varepsilon \cdot n)$. However, this causes the query complexity of P to be $\Omega(n)$, whereas we want our IPP to be doubly sub-linear. In the next section, we extend the foregoing IPP by applying it recursively, and show that we can get both query complexity of $o(\frac{1}{\varepsilon^2})$ for V, **and** poly-logarithmic query complexity for P.

3.3 The Actual dsIPP

In Section 3.2, we aim to achieve poly-logarithmic query complexity for the honest prover P, without compromising the query complexity of the verifier V. Unfortunately, for reasons explained at Section 3.2.2, P still has query complexity of $\Omega(n)$.

To solve this, we extend the IPP presented in Section 3.2 by applying it recursively: That is, after V selects at random $i \in [k]$, instead of directly computing the Hamming weight of x_i, both parties recursively run the protocol on x_i with the claimed Hamming weight σ_i, and do it for r rounds. Only in the base of the recursion, V computes the Hamming weight of the input (at this level), and accepts accordingly. This gives us an improvement in the query complexity of the verifier, as the length of the input after r rounds is $(\frac{n}{k^r})$, while the number of times the prover needs to run the approximator grows only to $k \cdot r$ (in every repetition). To make sure that the query complexity of V is $o(\frac{1}{\varepsilon^2})$, we set $k = n^{\frac{1}{r}}$.

Indeed, we now allow the prover to deviate from the claimed weight for r times: At each of the recursion levels, as well as in the base level. Thus, to make sure that soundness still holds (i.e. the total allowed deviation is not too big), we set the verifier's allowed deviation (i.e. δ) to be smaller (by a factor of r).

To make sure that completeness still holds, we first change the error probability parameter of the approximator. P now runs the approximator for a total of $t = O(\frac{k \cdot r}{\varepsilon})$ times, and so we set $\eta = 0.01 \cdot \frac{1}{t} = O(\frac{\varepsilon}{k \cdot r})$. In addition, we observe the following: Assume V interacts with P, with input x and weight parameter σ, such that $\mathtt{wt}(x) = \sigma$. Then, when we are not in the first recursion level, then the weight parameter provided to the protocol is the approximated weight provided by P in the previous recursion level. Therefore, the weight parameter in the current recursion level might deviate from the actual weight of the input in this level. Since P also performs weight approximations in the current recursion level, we need to account for both deviations. Details follow.

▶ **Protocol 1.** *dsIPP for the Hamming Weight Problem*

- ***Query Access Input:*** $x \in \{0,1\}^n$
- ***Deviation Parameter:*** $\varepsilon > 0$
- ***Other Parameters:*** *Weight* $\sigma \in \mathbb{N}$*, initial number of rounds* r_0 *and remaining number of rounds* r

1. *Set* $k = n^{\frac{1}{r_0}}$*,* $\delta = \frac{\varepsilon}{3r_0}$*, and* $\eta = O(\frac{\varepsilon}{k \cdot r_0})$*.*
2. *V: If* $r = 0$ *then accept iff* $|\mathtt{wt}(x) - \sigma| \leq \frac{\delta n}{2}$*.*
3. *P:*
   a. *For every* $i \in [k]$*, approximate* $\mathtt{wt}(x_i)$ *up to a deviation factor of* $\frac{\delta}{2}$ *with probability at least* $1 - \eta$*: Choose uniformly at random* $m = O(\frac{log(\frac{1}{\eta})}{\delta^2})$ *indices of* x_i *and set the approximation* σ_i *as the estimated fraction of indices* j *for which* $x_i[j] = 1$*.*
   b. *Send* $(\sigma_1, ..., \sigma_k)$ *to* V*.*
4. *V:*
   a. *Verify that* $|\sigma - \sum_i \sigma_i| \leq \delta n$*, otherwise reject.*
   b. *Choose uniformly at random* $i \in [k]$ *and send* i *to* P*.*
5. *Both parties recursively invoke the protocol with input* x_i*, deviation parameter* ε*, weight* σ_i*, and remaining number of rounds* $r - 1$*.*

Repeat the protocol for $O(\frac{1}{\varepsilon})$ *times, and accept iff* V *accepts in all repetitions.*

We show that Protocol 1 is a dsIPP for the Hamming Weight Problem: In Section 3.3.1, we show the correctness of the protocol, and in Section 3.3.2, we present the query complexity of the protocol.

3.3.1 Correctness

Completeness. Assume $\mathtt{wt}(x) = \sigma$ and V interacts with P. We observe that at any recursion level (but the base level), the following claim holds: If our input is x' and our weight parameter is σ' such that $|\mathtt{wt}(x') - \sigma'| \leq \frac{\delta|x'|}{2}$, then with probability at least $1 - \eta \cdot k$, the prover P sends $(\sigma'_1, ..., \sigma'_k)$ to V such that the following holds:

1. The verifier doesn't reject in Step $4(a)$ (i.e., $|\sigma' - \sum_i \sigma'_i| \leq \delta|x'|$).
2. The protocol is recursively invoked with some x'_i and σ'_i for which $|\mathtt{wt}(x'_i) - \sigma'_i| \leq \frac{\delta|x'|}{2}$.

The claims follows from the fact that P runs the approximator on every x'_i with deviation parameter $\frac{\delta}{2}$ and error probability parameter η. Then, since we start with $\mathtt{wt}(x) = \sigma$, with probability at least $1 - \eta \cdot k \cdot r_0$, the verifier doesn't reject in Step $4(a)$ in all the r_0 recursion levels. By the second item of the claim, the base of the recursion is also invoked with an input x' and weight parameter is σ' such that $|\mathtt{wt}(x') - \sigma'| \leq \frac{\delta|x'|}{2}$, and the verifier accepts. Since we repeat the protocol for $O(\frac{1}{\varepsilon})$ times, we get that the verifier accepts in all invocations with probability at least $1 - t \cdot \eta$ where $t = O(\frac{k \cdot r_0}{\varepsilon})$. Then, since we set $\eta = 0.01 \cdot \frac{1}{t} = O(\frac{\varepsilon}{k \cdot r_0})$, we get that V accepts with high constant probability in all invocations.

Soundness. Assume $|\mathtt{wt}(x) - \sigma| > \varepsilon n$. We observe that in each recursion level, we lose at most an additive factor of δ. Since we set $\delta = \frac{\varepsilon}{3r_0}$, we get that after r_0 rounds, we are still at distance at least $\frac{\varepsilon}{2}$ from a valid assertion. Thus, the verifier rejects with probability at least $\Omega(\varepsilon)$ *in each invocation*, and therefore rejects with high constant probability in at least one of the $O(\frac{1}{\varepsilon})$ invocations.

3.3.2 Computational Complexity

We present the query, communication and time complexity of Protocol 1. Let us start by analyzing the query complexity of V (denoted Q_V) and P (denoted Q_P):

- $\mathsf{Q}_V = O(\frac{n}{\varepsilon k^{r_0}}) = O(\frac{1}{\varepsilon})$, because we read a sub-sequence of the input of length $\frac{n}{k^{r_0}} = 1$ for $O(\frac{1}{\varepsilon})$ times.
- Q_P is the query complexity of the approximator times the number of calls P makes to the approximator in all repetitions:
  - The query complexity of the approximator is $O(\frac{r_0^2 \cdot \log(\frac{k \cdot r_0}{\varepsilon}))}{\varepsilon^2})$, because P invokes the approximator with error probability parameter $\eta = O(\frac{\varepsilon}{k \cdot r_0})$ and deviation parameter $\delta = O(\frac{\varepsilon}{r_0})$.
  - P calls the approximator for $k \cdot r_0$ times in each of the $O(\frac{1}{\varepsilon})$ repetitions.

  Since we set $k = n^{\frac{1}{r_0}}$, we get that $\mathsf{Q}_P = \tilde{O}(\frac{n^{\frac{1}{r_0}} \cdot r_0^3}{\varepsilon^3})$.

Hence, regardless of how we set r_0, the query complexity of the verifier is $O(\frac{1}{\varepsilon}) = o(\frac{1}{\varepsilon^2})$. Indeed, the larger we set r_0, the smaller Q_P gets, but this causes the round complexity to grow. Specifically, to minimize Q_P, we can set $r_0 = \log(n)$, and we get $\mathsf{Q}_P = \tilde{O}(\frac{\log^3(n)}{\varepsilon^3})$.

Now, the communication of Protocol 1 is $O(\frac{k \cdot r_0}{\varepsilon} \cdot \log(\frac{r_0}{\varepsilon}))$: In each of the $O(\frac{1}{\varepsilon})$ repetitions there are r_0 rounds, and in each round the prover P sends k weight approximations that can be represented by $O(\log(\frac{r_0}{\varepsilon}))$ bits (because the approximations are up to a factor of $\delta = O(\frac{\varepsilon}{r_0})$). Again, since we set $k = n^{\frac{1}{r_0}}$, we can set $r_0 = \log(n)$ and get poly-logarithmic communication complexity.

Lastly, the time complexity of the honest prover (denoted T_P) is the same as its query complexity (i.e., $\mathsf{T}_P = \mathsf{Q}_P$), whereas the time complexity of the verifier (denoted T_V) is $O(\frac{k \cdot r_0}{\varepsilon} \cdot \log(\frac{r_0}{\varepsilon}))$, because in each of the $O(\frac{1}{\varepsilon})$ repetitions there are r_0 rounds, and in each round the verifier performs simple calculations on $O(k)$ values that can be represented by $O(\log(\frac{r_0}{\varepsilon}))$ bits.

We conclude with the following theorem, that summarizes what we achieved in this section.

▶ **Theorem 10.** *For every function $r : \mathbb{N} \to \mathbb{N}$, there exists an $r(n)$-round dsIPP for the Hamming Weight Problem with $Q_V = O(\frac{1}{\varepsilon})$, $Q_P = \tilde{O}(\frac{n^{\frac{1}{r(n)}} \cdot r(n)^3}{\varepsilon^3})$ and communication complexity of $O(\frac{n^{\frac{1}{r(n)}} \cdot r(n) \cdot log(\frac{r(n)}{\varepsilon})}{\varepsilon})$. In addition, the time complxity of the dsIPP is $T_P = Q_P$ and $T_V = O(\frac{n^{\frac{1}{r(n)}} \cdot r(n) \cdot log(\frac{r(n)}{\varepsilon})}{\varepsilon})$.*

4 Tolerant dsIPP for ROOBPs of Constant Width

In this section, we present a tolerant dsIPP for ROOBPs of constant width. Before we review our plan in Section 4.2, let us recall some standard definitions related to ROOBPs.

4.1 Definitions

▶ **Definition 11** (ROOBP). *A **branching program (BP)** on n variables is a directed acyclic graph that has a unique source vertex (denoted s) with in-degree 0 and a unique sink vertex (denoted t) with out-degree 0. Each non-sink vertex is labeled by an index $i \in [n]$, and has 2 outgoing edges, which are labeled by either 0 or 1. An input $x \in \{0,1\}^n$ defines a walk on B starting at the source vertex, such that at every vertex labeled by $i \in [n]$, the step taken is on the edge labeled by x_i. The output of the branching program B on input $x \in \{0,1\}^n$, denoted $B(x)$, is defined as 1 if the walk reached the sink vertex, and 0 if it got "stuck" before reaching it (i.e., the walk reached vertex labeled by $i \in [n]$, and there was no outgoing edge labeled by x_i).*

*An **oblivious BP** is a BP in which the nodes are partitioned into levels, $L_0, ..., L_n$, and edges are going only from one level to nodes in the consecutive level. In addition, all the vertices of some level are associated with the same index. Therefore, we can say that the level itself is associated with some index. A **read-once oblivious BP (ROOBP)** is a BP in which no two levels are associated with the same index. An **ROOBP of constant width** w is an ROOBP in which every level has at most w vertices.*

From now on, let B be an ROOBP with n variables and width at most w.

▶ Remark 12. By the above, there's a 1-1 correspondence between an $s \rightsquigarrow t$ path in B and an **accepting** input for B. In that case, we say that the $s \rightsquigarrow t$ path is **associated** with the accepting input.

▶ Remark 13. We say that two vertices $u \in L_l$ and $v \in L_{l'}$ in B are connected only if they are connected via a **directed** path. If $l < l'$, we say that u is **forwards-connected** to v, and v is **backwards-connected** to u.

▶ Remark 14. Without loss of generality, we assume:

1. For every $i \in \{0, ..., n-1\}$, L_i is associated with the index i. Otherwise, we can change the input x to an input x' by reordering its indices accordingly.
2. B depends on the entire input. Otherwise, we can change the input to $x' \in \{0,1\}^{n'}$, and then B will depend on every index of x'.
3. There exists a path between the source and the sink of B; this is equivalent to assuming there exists an $x \in \{0,1\}^n$ accepted by B.

▶ **Definition 15** (Absolute and Relative Distance). *We denote the **absolute distance** between two strings $x, x' \in \{0,1\}^n$ by $\Delta(x, x') = |\{x[i] \neq x'[i] : i \in [n]|$, and their **relative distance** by $\overline{\Delta}(x, x') = \frac{\Delta(x,x')}{n}$. Assume there exist an accepting input for B, an ROOBP of length n.*

We denote the ***absolute distance*** *of a string $x \in \{0,1\}^n$ from B by $\Delta(x,B) = min\{\Delta(x,x') : x' \in \{0,1\}^n, B(x') = 1\}$. Similarly, the* ***relative distance*** *of x from B is denoted by $\overline{\Delta}(x,B) = min\{\overline{\Delta}(x,x') : x' \in \{0,1\}^n, B(x') = 1\}$.*

4.2 Our Plan

For a fixed ROOBP B of width w, proximity parameters $\varepsilon_f > \varepsilon_c \geq 0$, and oracle access to an input $x \in \{0,1\}^n$, we present an r-round IPP, with $r = \log(n)$ and with poly-logarithmic communication complexity, in which V uses query complexity of $O(\frac{1}{\varepsilon_f - \varepsilon_c})$, the honest prover P uses poly-logarithmic query complexity, and the following holds:

- If $\overline{\Delta}(B,x) \leq \varepsilon_c$, then when communicating with P, with probability at least $2/3$ the verifier V accepts x.
- If $\overline{\Delta}(B,x) > \varepsilon_f$, then when communicating with any prover P^*, with probability at least $2/3$ the verifier V rejects x.

We note that we use r to represent the number of (pairs) of message exchanges (and not the total number of messages sent).

Our plan is as follows: First, in Section 4.3, we present a standard tolerant IPP for the ROOBP Problem. This IPP (which is not doubly sub-linear) uses a divide-and-conquer approach, and in it the prover P partitions the ROOBP to a sequence of k sub-ROOBPs (see Definition 16) according to the accepting path associated with an accepting input that minimizes the distance to x in B (see Definition 17). Then, in Section 4.4, we present a warm-up towards the tolerant dsIPP: We amend the standard tolerant IPP by allowing P to partition the ROOBP according to an approximated path, and relaxing the accepting conditions of V in a way that doesn't compromise the completeness and soundness of the protocol. Unfortunately, in this IPP the prover P still has query complexity of $\Omega(n)$. Lastly, in Section 4.5, we present the actual tolerant dsIPP, which applies the IPP presented in Section 4.4 recursively, thus improving the query complexity of P without compromising the query complexity of V.

4.3 The Standard Tolerant IPP

Given some ROOBP B, and proximity parameters $\varepsilon_f > \varepsilon_c \geq 0$, we want to verify that $\overline{\Delta}(B,x) \leq \varepsilon_c$. In the following tolerant IPP, that adapts the non-tolerant IPP from [7], we decompose B to k consecutive equally-length sub-ROOBPs, $B_1, ..., B_k$. Therefore, we start with defining the notion of a sub-ROOBP.

▶ **Definition 16** (sub-ROOBP). *For $0 \leq i \leq j \leq n$, let $u \in L_i$ and $v \in L_j$. We define $B[u,v]$ as a* ***sub-ROOBP*** *of B, with the following properties:*

- *The source (resp., sink) vertex of $B[u,v]$ is u (resp., v).*
- *$B[u,v]$ is of length $j - i$.*
- *If $x \in \{0,1\}^n$ is the input for B, then $x[i+1,j] = x[i+1] \cdot x[i+2] \cdot \cdots x[j]$ is the input for $B[u,v]$.*

Indeed, there is more than one way we can decompose B to k consecutive equally-length sub-ROOBPs, because for each sub-ROOBP there could be up to w^2 possible source-sink pairs. Therefore, we continue with defining the notion of (what we consider) a valid decomposition of B to k sub-ROOBPs: A decomposition that doesn't cause us to "lose" any distance (i.e. the average distance of x from all the sub-ROOBPs in the decomposition is at least the distance of x from B). We can achieve this by enforcing the decomposition to follow some path associated with some accepting input for B, since for any accepting input x' we have $\overline{\Delta}(x,x') \geq \overline{\Delta}(x,B)$.

▶ **Definition 17** ((k,π)-Decomposition). *Let B be an ROOBP, and let π be an accepting path in B. The **(k,π)-decomposition** of B is a sequence of k sub-ROOBPs of B, denoted $\mathcal{B}_{k,\pi} = (B_1, ..., B_k)$, such that for every $i \in [k]$, the source of B_i is the $\frac{(i-1)\cdot n}{k}$'th vertex in π, and the sink of B_i is the $\frac{i\cdot n}{k}$'th vertex in π.*

Now, we observe that when we partition x to k consecutive equally-length parts, $x_1, ..., x_k$, the following holds:

- On the one hand, if x is ε_c-close to B, then for $(B_1, ..., B_k)$, the (k,π)-decomposition of B constructed according to an accepting input that minimizes the distance to x, it holds that $\sum_i \frac{\overline{\Delta}(B_i,x)}{k} \leq \varepsilon_c$.
- On the other hand, if $\overline{\Delta}(B,x) > \varepsilon_f$, then for any $(B_1, ..., B_k)$, a (k,π)-decomposition of B constructed according to some accepting path in B, it holds that $\sum_i \frac{\overline{\Delta}(B_i,x_i)}{k} > \varepsilon_f$.

Given the above, we consider the following IPP: The prover P finds the (k,π)-decomposition of B constructed according to an accepting input that minimizes the distance to x. Then, P checks the distance of x_i from each of the sub-ROOBPs in the decomposition, and sends V a succinct representation of the decomposition (for example, the sink of every sub-ROOBP), as well as the obtained distances. After V receives some decomposition $(B_1, ..., B_k)$ and claimed distances $(\hat{\varepsilon_1}, ..., \hat{\varepsilon_k})$, V verifies that $(B_1, ..., B_k)$ is a valid (k,π)-decomposition of B, and that the average of the claimed distances is at most ε_c. Lastly, V selects at random $i \in [k]$, and verifies that $\overline{\Delta}(B_i, x_i) = \hat{\varepsilon}_i$ (by reading the entire x_i).

Note that if x is ε_c-close to B and V interacts with P, then $\sum_i \frac{\hat{\varepsilon}_i}{k} \leq \varepsilon_c$, and for every $i \in [k]$, it holds that $\overline{\Delta}(B_i, x_i) = \hat{\varepsilon}_i$, and the verifier always accepts. However, if $\overline{\Delta}(B,x) > \varepsilon_f$, then for any valid decomposition $(B_1, ..., B_k)$ and claimed distances $(\hat{\varepsilon}_1, ..., \hat{\varepsilon}_k)$ sent by some prover P^* such that $\sum_i \frac{\hat{\varepsilon}_i}{k} \leq \varepsilon_c$, it holds that $\sum_i \frac{\overline{\Delta}(B_i,x_i)-\hat{\varepsilon}_i}{k} > \varepsilon_f - \varepsilon_c$. Then, the average deviation of the actual distance of x from B, from the claimed distance of x from B, translates to a deviation on a corresponding fraction of random i's, which guarantees that the verifier rejects with probability at least $\varepsilon_f - \varepsilon_c$. To increase the rejection probability, we can repeat this procedure for $O(\frac{1}{\varepsilon_f - \varepsilon_c})$ times.

We stress that V can verify the validity of a (k,π)-decomposition without any query access to the input x. Thus, the query complexity of the verifier is the length of a single x_i times the number of repetitions, which gives $O(\frac{n}{(\varepsilon_f - \varepsilon_c)\cdot k})$. However, to find an accepting input that minimizes the distance to x, the prover must access to the entire input x, which yields query complexity of $\Omega(n)$.

4.4 A Warm-Up towards a tolerant dsIPP

In this section, we aim to achieve poly-logarithmic query complexity for the honest prover P without compromising the query complexity of the verifier V: We amend the interaction described in Section 4.3 by allowing P to divide the ROOBP B according to an accepting input that *approximates* the distance of x to B. However, at the end of this section, we explain why P still has query complexity of $\Omega(n)$. In Section 4.5, we show how to solve this issue and present the actual tolerant dsIPP.

In the full version of this paper, we show the existence of a δ-distance-approximation algorithm for ROOBPs of constant width with query complexity that is independent of the length of the input, n. That is, we show there exists an algorithm that given an ROOBP B of width w, an input $x \in \{0,1\}^n$, deviation parameter $\delta > 0$ and error probability parameter $\eta > 0$, outputs $\hat{\varepsilon}$ such that with probability at least $1 - \eta$ it holds that $|\hat{\varepsilon} - \overline{\Delta}(B,x)| \leq \delta$, and this algorithms has query complexity of $(\frac{\log(\frac{1}{\eta})\cdot 2^w}{\delta})^{O(w)}$.

Based on this distance-approximation algorithm, we amend the interaction as follows: P performs distance approximation (with parameters δ and η we fix later) for every (k,π)-decomposition (constructed according to every accepting path π). Since P performs distance approximation also for the decomposition of B constructed according to an accepting input that minimizes the distance to x, P can just choose the the (k,π)-decomposition which yields the minimal (average) distance approximation. That is, P chooses $\mathcal{B}_{k,\pi^*} = (B_1^{\pi^*},...,B_k^{\pi^*})$ for which $\sum_i \frac{\hat{\varepsilon}_i^{\pi^*}}{k}$ is minimal.

Even though there could be many possible (k,π)-decompositions, the total number of sub-ROOBPs in all the (k,π)-decompositions is upper-bounded by $w^2 \cdot k$: For every $i \in [k]$, there are at most w^2 possible source-sink pairs for the i't sub-ROOBP. Thus, using the union bound, with probability at least $1 - w^2 \cdot k \cdot \eta$, all approximations deviate from the actual distance by at most δ, and the following holds:

- For every $i \in [k]$, it holds that $\overline{\Delta}(B_i^{\pi^*}, x_i) \leq \hat{\varepsilon}_i^{\pi^*} + \delta$.
- If $\overline{\Delta}(B,x) \leq \varepsilon_c$, then since P chooses the (k,π)-decomposition which yields the minimal (average) distance approximation, we get that $\sum_i \frac{\hat{\varepsilon}_i^{\pi^*}}{k} \leq \varepsilon_c + \delta$.

Lastly, we also change V's accepting conditions in the interaction, so that V allows the deviation. That is, after V receives $(B_1,...,B_k)$ and $(\hat{\varepsilon}_1,...,\hat{\varepsilon}_k)$, in addition to verifying that $(B_1,...,B_k)$ is a valid decomposition, V verifies that $\sum_i \frac{\hat{\varepsilon}_i}{k} \leq \varepsilon_c + \delta$, and after V chooses $i \in [k]$ uniformly at random, V verifies that $\overline{\Delta}(B_i, x_i) \leq \hat{\varepsilon}_i + \delta$ (by reading the entire x_i). Similar to the standard IPP, we repeat the protocol for $O(\frac{1}{\varepsilon_f - \varepsilon_c})$ times.

4.4.1 Correctness

We claim that the completeness of the protocol still holds (but with a completeness error). If V interacts with P, then for a single invocation of the protocol, using the union bound, with probability at least $1 - \eta \cdot k \cdot w^2$, all approximations deviate from the actual distances by at most δ. Then, if $\overline{\Delta}(B,x) \leq \varepsilon_c$, it holds that $\sum_i \frac{\hat{\varepsilon}_i^{\pi^*}}{k} \leq \varepsilon_c + \delta$, and for every $i \in [k]$ it holds that $\overline{\Delta}(B_i^{\pi^*}, x_i) \leq \hat{\varepsilon}_i^{\pi^*} + \delta$, and the verifier accepts. Since we invoke the protocol for $O(\frac{1}{\varepsilon_f - \varepsilon_c})$ times, then for $t = O(\frac{k \cdot w^2}{\varepsilon_f - \varepsilon_c})$, the verifier accepts in all invocations with probability at least $1 - \eta \cdot t$. Thus, we can set $\eta \leq 0.01 \cdot \frac{1}{t} = O(\frac{\varepsilon_f - \varepsilon_c}{k \cdot w^2})$, and the verifier accepts with high constant probability in all invocations.

To show the soundness of the protocol also holds, assume $\overline{\Delta}(B,x) > \varepsilon_f$, and let $(B_1,...,B_k)$ be a decomposition that P^* sends, along with the respective claimed distance approximations $(\hat{\varepsilon}_1,...,\hat{\varepsilon}_k)$. If the decomposition is valid, then $\sum_i \frac{\overline{\Delta}(B_i,x_i)}{k} > \varepsilon_f$. Thus, if $\sum_i \frac{\hat{\varepsilon}_i}{k} \leq \varepsilon_c + \delta$, we get that $\sum_i \frac{\overline{\Delta}(B_i,x_i) - \hat{\varepsilon}_i}{k} > \varepsilon_f - \varepsilon_c - \delta$. Then, the average deviation of the actual distance of x from B, from the claimed distance of x from B, beyond the allowed deviation, translates to a deviation on a corresponding fraction of random i's. Since the verifier rejects if $\overline{\Delta}(B_i,x_i) > \hat{\varepsilon}_i + \delta$, we can set $\delta = \frac{\varepsilon_f - \varepsilon_c}{3}$, and infer that the verifier still rejects with probability at least $\Omega(\varepsilon_f - \varepsilon_c)$ in a single invocation. Since we invoke the protocol for $O(\frac{1}{\varepsilon_f - \varepsilon_c})$ times, we get that the verifier rejects with high constant probability in at least one of the invocations.

4.4.2 Query Complexity

Now, let us analyze the query complexity of V and P:

- Because V can verify that the (k,π)-decomposition that P sends is valid without any query access to the input x, the query complexity of V is $O(\frac{n}{(\varepsilon_f - \varepsilon_c) \cdot k})$: V reads a sub-sequence of the input of length $\frac{n}{k}$ for $O(\frac{1}{\varepsilon_f - \varepsilon_c})$ repetitions.

- The query complexity of P is $k \cdot (\frac{\log(\frac{k}{\varepsilon_f - \varepsilon_c}) \cdot 2^w}{\varepsilon_f - \varepsilon_c})^{O(w)}$, because the distance approximation algorithm with deviation parameter $\delta = O(\varepsilon_f - \varepsilon_c)$ and error probability parameter $\eta = O(\frac{\varepsilon_f - \varepsilon_c}{w^2 \cdot k})$ has query complexity of $(\frac{\log(\frac{k}{\varepsilon_f - \varepsilon_c}) \cdot 2^w}{\varepsilon_f - \varepsilon_c})^{O(w)}$, and P invokes the algorithm for $w^2 \cdot k$ times in each of the $O(\frac{1}{\varepsilon_f - \varepsilon_c})$ repetitions.

Since we can δ-approximate $\overline{\Delta}(B, x)$ without any interaction using query complexity $(\frac{2^w}{\delta})^{O(w)}$, we can perform tolerant $(\varepsilon_f, \varepsilon_c)$-test for ROOBPs using query complexity $(\frac{2^w}{\varepsilon_f - \varepsilon_c})^{O(w)}$. Therefore, for the IPP to be meaningful, we want the query complexity of V to be $o((\frac{2^w}{\varepsilon_f - \varepsilon_c})^{O(w)})$. To achieve this in our setting, we must require $k = \omega((\frac{\varepsilon_f - \varepsilon_c}{2^w})^{O(w)}) \cdot n)$. However, this causes the query complexity of P to be $\Omega(n)$, whereas we want our IPP to be doubly sub-linear. In the next section, we extend the foregoing IPP by applying it recursively, and show that we can get both query complexity of $o((\frac{2^w}{\varepsilon_f - \varepsilon_c})^{O(w)})$ for V, **and** poly-logarithmic query complexity for P.

4.5 The Actual dsIPP

In Section 4.4, we aim to achieve poly-logarithmic query complexity for the honest prover P, without compromising the query complexity of the verifier V. Unfortunately, for reasons explained at Section 4.4.2, the prover P still has query complexity of $\Omega(n)$.

To solve this, we extend the IPP presented in Section 4.4 by applying it recursively: That is, after V selects at random $i \in [k]$, instead of directly checking the distance of x_i from B_i, both parties recursively run the protocol on x_i with B_i, and do it for r rounds. Only in the base of the recursion, assuming the input is x' and the ROOBP is B', V checks the distance of x' from B' and accepts accordingly. This gives us an improvement in the query complexity of the verifier, as the length of the input after r rounds is $(\frac{n}{k^r})$, while the number of times the prover needs to run the distance approximation algorithm grows only to $w^2 \cdot k \cdot r$ (in every repetition). In order to make the query complexity of V independent of the input length, we set $k = n^{\frac{1}{r}}$.

Now, we observe the following: Assume V interacts with P, with input x and ROOBP B, such that $\overline{\Delta}(B, x) \leq \varepsilon_c$. Then, since we are only guaranteed that $\sum_i \frac{\hat{\varepsilon}_i}{k} \leq \varepsilon_c + \delta$, when both parties recursively apply the protocol with input x_i and ROOBP B_i, we can't guarantee that $\overline{\Delta}(B_i, x_i) \leq \varepsilon_c$ (or even guarantee that $\overline{\Delta}(B_i, x_i) \leq \varepsilon_c + \delta$).

Therefore, we amend the protocol as follows: In addition to the input x and the ROOBP B, we add a claimed distance parameter, $\hat{\varepsilon}$, that represents P's claim regarding the distance of x from B. In the first recursion level, we set $\hat{\varepsilon} = \varepsilon_c$, since P claims that $\overline{\Delta}(B, x) \leq \varepsilon_c$. Then, at each recursion level, assume our input x', our ROOBP is B', our (new) claimed distance parameter is $\hat{\varepsilon}'$, and P sends the decomposition $(B_1, ..., B_k)$ and the respective approximations $(\hat{\varepsilon}'_1, ..., \hat{\varepsilon}'_k)$. Then, V verifies that $(B_1, ..., B_k)$ is a valid decomposition and that $\sum_i \frac{\hat{\varepsilon}'_i}{k} \leq \hat{\varepsilon}' + \delta$ (i.e., we replace ε_c in the previous condition with the new parameter $\hat{\varepsilon}'$). Lastly, for some $i \in [k]$ both parties recursively run the protocol on x'_i with B'_i and the claimed distance $\hat{\varepsilon}'_i$, and at the base of the recursion, V verifies that the distance of the input from the ROOBP does not exceed the **claimed distance** by more than δ.

Now, on every recursion level, P only claims that the average distance doesn't grow too much **with respect to the approximated distance in the previous recursion level**. Since P uses a distance approximation algorithm with some error probability parameter $\eta > 0$, and chooses the (k, π)-decomposition of B that yields the minimal average distance approximation from x, the claim holds for all recursion levels with probability at least $1 - \eta \cdot k \cdot w^2 \cdot r_0$, and at the base level, the verifier accepts. Since P invokes the protocol for

$O(\frac{1}{\varepsilon_f-\varepsilon_c})$ times, then for $t = O(\frac{w^2\cdot k\cdot r}{\varepsilon_f-\varepsilon_c})$, the verifier accepts in all invocations with probability at least $1-\eta\cdot t$. Thus, we can set $\eta = 0.01\cdot\frac{1}{t} = O(\frac{\varepsilon_f-\varepsilon_c}{w^2\cdot k\cdot r})$, and the verifier accepts with high constant probability in all invocations. Lastly, we note that at each recursion level, we want V to account for two possible deviations: In the approximation that P provides for the previous recursion level (i.e. $\hat{\varepsilon}$), and in the approximations that P performs in the current recursion level (i.e. $(\hat{\varepsilon}_1,...,\hat{\varepsilon}_k)$).

For soundness, we observe that V allows the prover to exceed from the initial claimed distance (i.e., ε_c) by at most δ for r times: At each of the recursion levels, as well as in the base level. Thus, to make sure that soundness still holds (i.e. V does not allow the prover to exceed too much in distance), we set δ to be smaller (by a factor of r). Details follow.

▶ **Protocol 2.** *Tolerant dsIPP for the ROOBP Problem*

- ***Query Access Input:*** $x\in\{0,1\}^n$
- ***Proximity Parameters:*** $\varepsilon_f > \varepsilon_c \geq 0$
- ***Massive Parameter:*** *ROOBP B of width w*
- ***Other Parameters:*** *Claimed distance $\hat{\varepsilon}$ (initialized to ε_c in the first round), initial number of rounds r_0 and remaining number of rounds r*

1. *Set $k = n^{\frac{1}{r_0}}$, $\delta = \frac{\varepsilon_f-\varepsilon_c}{3r_0}$, and $\eta = O(\frac{\varepsilon_f-\varepsilon_c}{r_0\cdot k\cdot w^2})$.*
2. *V: If $r=0$ then accept iff $\overline{\Delta}(B,x)\leq\hat{\varepsilon}+\frac{\delta}{2}$.*
3. *P:*
   a. *For every $i\in[k]$, and for every $u\in L_{\frac{(i-1)\cdot n}{k}}$ and $v\in L_{\frac{i\cdot n}{k}}$, approximate the distance of x_i from $B[u,v]$ with deviation parameter $\frac{\delta}{2}$ and error probability parameter η.*
   b. *For every (k,π)-decomposition of B, $\mathcal{B}_{k,\pi} = (B_1^\pi,...,B_k^\pi)$, let $(\hat{\varepsilon}_1^\pi,...,\hat{\varepsilon}_k^\pi)$ be the respective distance approximations obtained in the previous step.*
   c. *Find a decomposition $\mathcal{B}_{k,\pi^*}$ for which $\sum_i\hat{\varepsilon}_i^{\pi^*}$ is minimal, and send (a succinct representation of) $\mathcal{B}_{k,\pi^*}$, along with $(\hat{\varepsilon}_1^{\pi^*},...,\hat{\varepsilon}_k^{\pi^*})$, to V.*
4. *V:*
   a. *Verify that $\mathcal{B}_{k,\pi^*}$ is a valid (k,π)-decomposition, and that $\sum_i\frac{\hat{\varepsilon}_i^{\pi^*}}{k}\leq\hat{\varepsilon}+\delta$. Otherwise, reject.*
   b. *Choose uniformly at random $i\in[k]$ and send i to P.*
5. *Both parties recursively invoke the protocol with input x_i, ROOBP $B_i^{\pi^*}$, claimed distance $\hat{\varepsilon}_i^{\pi^*}$, and remaining number of rounds $r-1$.*

Initiate the protocol with $\hat{\varepsilon}=\varepsilon_c$, repeat the protocol for $O(\frac{1}{\varepsilon_f-\varepsilon_c})$ times, and accept iff V accepts in all repetitions.

▶ Remark 18. P can efficiently find a decomposition in Step 3(c) as follows: Construct a graph $G=(V,E)$ with V being all the vertices of B in the levels $(L_0,L_{\frac{n}{k}},L_{\frac{2n}{k}}...,L_{\frac{k\cdot n}{k}})$, and $u,v\in V$ are connected in G with edge-weight $\hat{\varepsilon}_{u,v}$ if P performs distance approximation for $B[u,v]$ in Step 3(a) and it results in $\hat{\varepsilon}_{u,v}$. Then, every (k,π)-decomposition considered in Step 3(b) has a corresponding $s\rightsquigarrow t$ path in G, and the weight of the $s\rightsquigarrow t$ path equals the sum of distance approximations for the sub-ROOBPs in the decomposition. Thus, P can choose the decomposition with the shortest $s\rightsquigarrow t$ path in G.

We show that Protocol 2 is a dsIPP for ROOBPs: In Section 4.5.1, we show the correctness of the protocol, and in Section 4.6.1, we present the query complexity of the protocol.

4.5.1 Correctness

Completeness. Assume $\overline{\Delta}(B,x) \leq \varepsilon_c$ and V interacts with P. We observe that at any recursion level (but the base level), the following claim holds: If our input is x', our ROOBP is B' and our claimed distance parameter is $\hat{\varepsilon}'$ such that $\overline{\Delta}(B',x') \leq \hat{\varepsilon}' + \frac{\delta}{2}$, then with probability at least $1 - \eta \cdot k \cdot w^2$, the prover P sends $(B'_1, ..., B'_k)$ and $(\hat{\varepsilon}'_1, ..., \hat{\varepsilon}'_k)$ to V such that the following holds:

1. The verifier doesn't reject in Step $4(a)$ (i.e., $\sum_i \frac{\hat{\varepsilon}'_i}{k} \leq \hat{\varepsilon}' + \delta$).
2. The protocol is recursively invoked with some x'_i, B'_i and $\hat{\varepsilon}'_i$ for which $\overline{\Delta}(B'_i, x'_i) \leq \hat{\varepsilon}'_i + \frac{\delta}{2}$.

The claims follows from the fact that P runs the distance approximation algorithm with deviation parameter $\frac{\delta}{2}$ and with error probability parameter η. Since we start with $\overline{\Delta}(B,x) \leq \varepsilon_c = \hat{\varepsilon}$, then with probability at least $1 - \eta \cdot k \cdot w^2 \cdot r_0$, the verifier doesn't reject in Step $4(a)$ in all the r_0 recursion levels. By the second item of the claim, the base of the recursion is also invoked with an input x', ROOBP B' and claimed distance parameter $\hat{\varepsilon}'$ such that $\overline{\Delta}(B',x') \leq \hat{\varepsilon}' + \frac{\delta}{2}$, and the verifier accepts. Since we repeat the protocol for $O(\frac{1}{\varepsilon_f - \varepsilon_c})$ times, we get that for $t = O(\frac{k \cdot w^2 \cdot r_0}{\varepsilon_f - \varepsilon_c})$, the verifier accepts in all invocations with probability at least $1 - \eta \cdot t$. Then, since we set $\eta = 0.01 \cdot \frac{1}{t} = O(\frac{\varepsilon_f - \varepsilon_c}{k \cdot w^2 \cdot r_0})$, we get that V accepts with high constant probability in all invocations.

Soundness. Assume $\overline{\Delta}(B,x) > \varepsilon_f$. We observe that in each round, we lose at most an additive factor of δ in distance (i.e. we allow the prover to "exceed" the claimed distance up to an additive δ factor). Since we start with claimed distance ε_c, and since we set $\delta = \frac{\varepsilon_f - \varepsilon_c}{3r_0}$, we get that after r_0 rounds, we are still at distance at least $\frac{\varepsilon_f - \varepsilon_c}{2}$ from a valid assertion. Hence, the verifier rejects with probability at least $\Omega(\varepsilon_f - \varepsilon_c)$ *in each invocation*, and rejects with high constant probability in at least one of the $O(\frac{1}{\varepsilon_f - \varepsilon_c})$ invocations.

4.6 Computational Complexity (of Protocol 2)

The query and communication complexity of our tolerant dsIPP are our main complexity measures. Indeed, the communication complexity is $O(\frac{k \cdot \log(w \cdot n) \cdot r_0}{\varepsilon_f - \varepsilon_c})$, because in each of the $O(\frac{1}{\varepsilon_f - \varepsilon_c})$ repetitions there are r_0 rounds, and in each round the prover P sends k vertices and k distance approximations to V that can be represented by $O(k \cdot \log(w \cdot n))$ bits. Since we set $k = n^{\frac{1}{r_0}}$, we can set $r_0 = \log(n)$ and get poly-logarithmic communication complexity.

We continue in Section 4.6.1, where we calculate the query complexity of both the verifier and the prover. We show that if we set $r_0 = \log(n)$, then V uses query complexity of $O(\frac{1}{\varepsilon_f - \varepsilon_c})$, and the honest prover P uses poly-logarithmic query complexity.

We finish by calculating the time complexity of our tolerant dsIPP. We show that, if we set $r_0 = \log(n)$, and assume both V and P have black-box access to two procedures that refer to the input ROOBP (see Section 4.6.2 for more details), then the verifier V has time complexity of $O(\frac{\log^2(n)}{\varepsilon_f - \varepsilon_c})$. In addition, if the honest prover P uses a distance approximation algorithm for ROOBPs with time complexity T_A, then P has time complexity of $O(\frac{\log^2(n) \cdot \mathsf{T}_A}{\varepsilon_f - \varepsilon_c})$.

4.6.1 Query Complexity

Let us analyze the query complexity of V (denoted Q_V) and P (denoted Q_P):

- Because V can verify that the (k,π)-decomposition P sends in every round is valid without any query access to the input x, we have $\mathsf{Q}_V = O(\frac{n}{(\varepsilon_f - \varepsilon_c) \cdot k^{r_0}}) = O(\frac{1}{\varepsilon_f - \varepsilon_c})$: V reads a sub-sequence of the input of length $\frac{n}{k^{r_0}} = 1$ for $O(\frac{1}{\varepsilon_f - \varepsilon_c})$ times.

- P uses a distance approximation algorithm A with deviation parameter $\delta = O(\frac{\varepsilon_f - \varepsilon_c}{r_0})$ and error probability parameter $\eta = O(\frac{\varepsilon_f - \varepsilon_c}{r_0 \cdot k \cdot w^2})$, with query complexity $\mathsf{Q}_A(\delta, \eta, w)$. Then, the query complexity of Q_P is $\mathsf{Q}_A(\delta, \eta, w)$ times the number of calls P makes to A in all the repetitions. Since P calls the approximator for $w^2 \cdot k \cdot r_0$ times in each of the $O(\frac{1}{\varepsilon_f - \varepsilon_c})$ repetitions, and we set $k = n^{\frac{1}{r_0}}$, we get that $\mathsf{Q}_P = O(\frac{w^2 \cdot n^{\frac{1}{r_0}} \cdot r_0}{\varepsilon_f - \varepsilon_c} \cdot \mathsf{Q}_A(\delta, \eta, w))$.

Hence, regardless of how we set r_0, the query complexity of the verifier is $O(\frac{1}{\varepsilon_f - \varepsilon_c})$. Indeed, the larger we set r_0, the smaller Q_P gets, but this causes the round complexity to grow. To minimize Q_P, we can set $r_0 = \log(n)$, and we get $\mathsf{Q}_P = O(\frac{w^2 \cdot \log(n)}{\varepsilon_f - \varepsilon_c} \cdot \mathsf{Q}_A(\delta, \eta, w))$.

In the full version of this paper, we show the existence of a distance approximation algorithm with query complexity of $\mathsf{Q}_A(\delta, \eta, w) = (\frac{\log(\frac{1}{\eta}) \cdot 2^w}{\delta})^{O(w)}$. If P uses this distance approximation algorithm, we get $\mathsf{Q}_P = \tilde{O}(n^{\frac{1}{r_0}}) \cdot (\frac{r_0 \cdot 2^w}{\varepsilon_f - \varepsilon_c})^{O(w)}$. Again, we can set $r_0 = \log(n)$, and if we assume w is constant, we get poly-logarithmic query complexity.

4.6.2 Time Complexity

Indeed, assuming a standard access to the input ROOBP B (i.e. the "Massive Parameter"), the time complexity of both the verifier (denoted T_V) and the honest prover (denoted T_P) is $\Omega(n)$:

- Given a sub-sequence of vertices $(v_0, ..., v_k)$ in B, the verifier V has to verify that every v_i is in the $\frac{i \cdot k}{n}$'th level of B, and that $v_i \rightsquigarrow v_{i+1}$. Since $v_0 \in L_0$ and $v_k \in L_n$, this requires accessing the entire ROOBP, thus yields time complexity of $\Omega(n)$.
- P approximates the distance of sub-parts of x from sub-ROOBPs of B of length $\frac{n}{k}$ for $w^2 \cdot k$ times. Using the distance approximation algorithm presented in the full version of this paper, this also yields time complexity of $\Omega(n)$.

However, we can take an alternative approach in the analysis: Assume both V and P have black-box access to the following two procedures, that refer to the structure of the input ROOBP, and are independent of the specific input x:

1. Given two vertices u and v, determine whether $u \rightsquigarrow v$ in B.
2. Given $i \in \{0, ..., n\}$, list the (up to w) vertices that are in level i of B.

In addition, assume P uses a distance approximation algorithm A with deviation parameter $\delta = O(\frac{\varepsilon_f - \varepsilon_c}{r_0})$ and error probability parameter $\eta = O(\frac{\varepsilon_f - \varepsilon_c}{r_0 \cdot k \cdot w^2})$, with time complexity $\mathsf{T}_A(\delta, \eta, n, w)$ (where n is the length of the input and w is the width of the ROOBP). Now, for $i \in [r_0]$, we can analyze the time complexity of V and P *in the i'th round of the protocol*:

- V calls the first two procedures for $O(k)$ times, obtains values that can be represented by $O(k \cdot \log(w \cdot n))$ bits, and performs simple calculations on the obtained values.
- P calls the first two procedures for k times, and calls the distance approximation algorithm for at most $w^2 \cdot k$ times on inputs of length $\frac{n}{k^i}$. By doing the foregoing, P obtains values that can be represented by $O(w^2 \cdot k \cdot \log(w \cdot n))$ bits, and performs simple calculations on the obtained values. This yields time complexity of $O(w^2 \cdot k \cdot (\log(w \cdot n) + \mathsf{T}_A(\delta, \eta, \frac{n}{k^i}, w)))$.

Since V and P perform the foregoing for every $i \in [r_0]$ in each of the $O(\frac{1}{\varepsilon_f - \varepsilon_c})$ invocations of the protocol, and since we set $k = n^{\frac{1}{r_0}}$, we get that:

- $\mathsf{T}_V = O(\frac{n^{\frac{1}{r_0}} \cdot r_0 \cdot \log(w \cdot n)}{\varepsilon_f - \varepsilon_c})$
- $\mathsf{T}_P = O(\frac{w^2 \cdot n^{\frac{1}{r_0}} \cdot (r_0 \cdot \log(w \cdot n) + \sum_{i=1}^{r_0} \mathsf{T}_A(\delta, \eta, n^{\frac{r_0 - i}{r_0}}, w))}{\varepsilon_f - \varepsilon_c})$

To minimize the time complexity, we can set $r_0 = \log(n)$, and assuming w is constant, we get $\mathsf{T}_V = O(\frac{\log^2(n)}{\varepsilon_f - \varepsilon_c})$ and $\mathsf{T}_P = O(\frac{\log^2(n)+\log(n)\cdot \mathsf{T}_A(\delta,\eta,n,w)}{\varepsilon_f - \varepsilon_c})$. We conclude with the following theorem and corollary.

▶ **Theorem 19.** *For every function $r : \mathbb{N} \to \mathbb{N}$, there exists an $r(n)$-round tolerant dsIPP for ROOBPs of width w with proximity parameters $\varepsilon_f > \varepsilon_c \geq 0$. In the tolerant dsIPP, the honest prover P uses a distance approximation algorithm A with deviation parameter $\delta = O(\frac{\varepsilon_f - \varepsilon_c}{r(n)})$ and error probability parameter $\eta = O(\frac{\varepsilon_f - \varepsilon_c}{r(n)\cdot n^{\frac{1}{r(n)}}\cdot w^2})$ such that the query complexity of A is $\mathsf{Q}_A(\delta,\eta)$ and the time complexity of A is $\mathsf{T}_A(\delta,\eta,n,w)$. Then, the computational complexity of the tolerant dsIPP is as follows:*

- ***Query Complexity**: $\mathsf{Q}_V = O(\frac{1}{\varepsilon_f - \varepsilon_c})$ and $\mathsf{Q}_P = O(\frac{w^2\cdot n^{\frac{1}{r(n)}}\cdot r(n)}{\varepsilon_f - \varepsilon_c} \cdot \mathsf{Q}_A(\delta,\eta))$.*
- ***Communication Complexity**: $O(\frac{n^{\frac{1}{r(n)}}\cdot r(n)\cdot log(w\cdot n)}{\varepsilon_f - \varepsilon_c})$.*
- ***Time Complexity**: If both V and P have black-box access to procedures that refer to the structure of the input ROOBP, then:*
  - $\mathsf{T}_V = O(\frac{n^{\frac{1}{r(n)}}\cdot r(n)\cdot log(w\cdot n)}{\varepsilon_f - \varepsilon_c})$
  - $\mathsf{T}_P = O(\frac{w^2\cdot n^{\frac{1}{r(n)}}\cdot (r(n)\cdot log(w\cdot n) + \sum_{i=1}^{r(n)} \mathsf{T}_A(\delta,\eta,n^{\frac{r(n)-i}{r(n)}},w))}{\varepsilon_f - \varepsilon_c})$

In the full version of this paper, we show the existence of a distance approximation algorithm with time complexity of $\mathsf{T}_A(\delta,\eta,n,w) = (\frac{\log(\frac{1}{\eta})\cdot 2^w}{\delta})^{O(w)}\cdot n$. If P uses this distance approximation algorithm, we get the following corollary.

▶ **Corollary 20.** *For every function $r : \mathbb{N} \to \mathbb{N}$, there exists an $r(n)$-round tolerant dsIPP for ROOBPs of width w with $\mathsf{Q}_V = O(\frac{1}{\varepsilon_f - \varepsilon_c})$, $\mathsf{Q}_P = \tilde{O}(n^{\frac{1}{r(n)}})\cdot(\frac{r(n)\cdot 2^w}{\varepsilon_f - \varepsilon_c})^{O(w)}$, and communication complexity of $O(\frac{n^{\frac{1}{r(n)}}\cdot r(n)\cdot log(w\cdot n)}{\varepsilon_f - \varepsilon_c})$. If both V and P have black-box access to procedures that refer to the structure of the input ROOBP, then $\mathsf{T}_V = O(\frac{n^{\frac{1}{r(n)}}\cdot r(n)\cdot log(w\cdot n)}{\varepsilon_f - \varepsilon_c})$ and $\mathsf{T}_P = (\frac{r(n)\cdot 2^w\cdot log(n)}{\varepsilon_f - \varepsilon_c})^{O(w)}\cdot n$.*

References

1. Ran Canetti, Guy Even, and Oded Goldreich. Lower bounds for sampling algorithms for estimating the average. *Inf. Process. Lett.*, 53(1):17–25, 1995. doi:10.1016/0020-0190(94)00171-T.
2. Alessandro Chiesa and Tom Gur. Proofs of proximity for distribution testing. In Anna R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, volume 94 of *LIPIcs*, pages 53:1–53:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPICS.ITCS.2018.53.
3. Funda Ergün, Ravi Kumar, and Ronitt Rubinfeld. Fast approximate probabilistically checkable proofs. *Inf. Comput.*, 189(2):135–159, 2004. doi:10.1016/J.IC.2003.09.005.
4. Oded Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017. doi:10.1017/9781108135252.
5. Oded Goldreich. On doubly-efficient interactive proof systems. *Found. Trends Theor. Comput. Sci.*, 13(3):158–246, 2018. doi:10.1561/0400000084.
6. Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998. doi:10.1145/285055.285060.
7. Oded Goldreich, Tom Gur, and Ron D. Rothblum. Proofs of proximity for context-free languages and read-once branching programs. *Inf. Comput.*, 261:175–201, 2018. doi:10.1016/J.IC.2018.02.003.

8 Oded Goldreich and Dana Ron. A sublinear bipartiteness tester for bounded degree graphs. *Comb.*, 19(3):335–373, 1999. doi:10.1007/S004930050060.

9 Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. *Algorithmica*, 32(2):302–343, 2002. doi:10.1007/S00453-001-0078-7.

10 Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: Interactive proofs for muggles. *J. ACM*, 62(4):27:1–27:64, 2015. doi:10.1145/2699436.

11 Shafi Goldwasser, Guy N. Rothblum, Jonathan Shafer, and Amir Yehudayoff. Interactive proofs for verifying machine learning. In James R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8, 2021, Virtual Conference*, volume 185 of *LIPIcs*, pages 41:1–41:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICS.ITCS.2021.41.

12 Tom Gur, Mohammad Mahdi Jahanara, Mohammad Mahdi Khodabandeh, Ninad Rajgopal, Bahar Salamatian, and Igor Shinkar. On the power of interactive proofs for learning. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, pages 1063–1070. ACM, 2024. doi:10.1145/3618260.3649784.

13 Tom Gur, Yang P. Liu, and Ron D. Rothblum. An exponential separation between MA and AM proofs of proximity. *Comput. Complex.*, 30(2):12, 2021. doi:10.1007/S00037-021-00212-3.

14 Tal Herman and Guy N. Rothblum. Verifying the unseen: interactive proofs for label-invariant distribution properties. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 – 24, 2022*, pages 1208–1219. ACM, 2022. doi:10.1145/3519935.3519987.

15 Tal Herman and Guy N. Rothblum. Doubley-efficient interactive proofs for distribution properties. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 743–751. IEEE, 2023. doi:10.1109/FOCS57990.2023.00049.

16 Ilan Newman. Testing membership in languages that have small width branching programs. *SIAM J. Comput.*, 31(5):1557–1570, 2002. doi:10.1137/S009753970038211X.

17 Michal Parnas, Dana Ron, and Ronitt Rubinfeld. Tolerant property testing and distance approximation. *J. Comput. Syst. Sci.*, 72(6):1012–1042, 2006. doi:10.1016/J.JCSS.2006.03.002.

18 Pavel Pudlák. A lower bound on complexity of branching programs (extended abstract). In Michal Chytil and Václav Koubek, editors, *Mathematical Foundations of Computer Science 1984, Praha, Czechoslovakia, September 3-7, 1984, Proceedings*, volume 176 of *Lecture Notes in Computer Science*, pages 480–489. Springer, 1984. doi:10.1007/BFB0030331.

19 Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. *SIAM J. Comput.*, 50(3), 2021. doi:10.1137/16M1096773.

20 Guy N. Rothblum and Ron D. Rothblum. Batch verification and proofs of proximity with polylog overhead. In Rafael Pass and Krzysztof Pietrzak, editors, *Theory of Cryptography – 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part II*, volume 12551 of *Lecture Notes in Computer Science*, pages 108–138. Springer, 2020. doi:10.1007/978-3-030-64378-2_5.

21 Guy N. Rothblum, Salil P. Vadhan, and Avi Wigderson. Interactive proofs of proximity: delegating computation in sublinear time. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 793–802. ACM, 2013. doi:10.1145/2488608.2488709.

22 Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM J. Comput.*, 25(2):252–271, 1996. doi:10.1137/S0097539793255151.

23 Hadar Strauss. Emulating computationally sound public-coin ipps in the pre-coordinated model. *Electronic Colloquium on Computational Complexity (ECCC)*, TR24-131, 2024. URL: https://eccc.weizmann.ac.il/report/2024/131/.