Doubly Sub-linear Interactive Proofs of Proximity

Noga Amir* Weizmann Institute

Oded Goldreich† Weizmann Institute

Guy N. Rothblum‡ Apple

September 25, 2024

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. 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.

*Email: noga.amir@weizmann.ac.il.

†Email: oded.goldreich@weizmann.ac.il.

‡Email: rothblum@alum.mit.edu.

Contents

1 Introduction **1**
- 1.1 Our focus: query-efficient generation of proofs of proximity . . . 1
- 1.2 Our Main Results . . . 2
  - 1.2.1 Protocol for Read-Once Oblivious Branching Programs . . . 2
  - 1.2.2 Protocol for Hamming Weight . . . 4
  - 1.2.3 Protocol for PERM . . . 5
  - 1.2.4 Protocol for Bipartiteness . . . 6
- 1.3 Further Related Work . . . 8
- 1.4 Technical Overview . . . 8

2 Preliminaries and Definitions **10**

3 ds-IPP for the Hamming Weight Problem **11**
- 3.1 The Standard IPP . . . 11
- 3.2 A Warm-Up towards a ds-IPP . . . 11
  - 3.2.1 Correctness . . . 12
  - 3.2.2 Query Complexity . . . 12
- 3.3 The Actual ds-IPP . . . 13
  - 3.3.1 Correctness . . . 14
  - 3.3.2 Computational Complexity . . . 14

4 Tolerant ds-IPP for ROOBPs of Constant Width **15**
- 4.1 Definitions . . . 15
- 4.2 Our Plan . . . 16
- 4.3 The Standard Tolerant IPP . . . 17
- 4.4 A Warm-Up towards a tolerant ds-IPP . . . 18
  - 4.4.1 Correctness . . . 19
  - 4.4.2 Query Complexity . . . 19
- 4.5 The Actual ds-IPP . . . 19
  - 4.5.1 Correctness . . . 21
- 4.6 Computational Complexity (of Protocol 4.8) . . . 22
  - 4.6.1 Query Complexity . . . 22
  - 4.6.2 Time Complexity . . . 23

A δ-Distance-Approximation Algorithm for an ROOBP of Constant Width **28**
- A.1 Introduction . . . 28
- A.2 δ-Distance-Approximation Algorithm for a ROOBP of width 1 . . . 28
- A.3 δ-Distance-Approximation Algorithm for a w-sparse ROOBP . . . 29
  - A.3.1 Definitions . . . 29
  - A.3.2 Finding of a δ-Good Pair . . . 31
  - A.3.3 The Actual Algorithm . . . 33
- A.4 δ-Distance-Approximation Algorithm for a (α, β, w)-locally-sparse ROOBP . . . 35
- A.5 δ-Distance-Approximation Algorithm for an ROOBP of Constant Width . . . 39
  - A.5.1 Choosing the Dissection Levels of the Final Decomposition . . . 39
  - A.5.2 Choosing the Sources and Sinks of the Final Decomposition . . . 40
  - A.5.3 Choosing sub-ROOBPs from the Final Decomposition . . . 42
  - A.5.4 The Actual Algorithm . . . 42
- A.6 Computational Complexity (of Algorithm A.33) . . . 44
  - A.6.1 Query Complexity . . . 45
  - A.6.2 Time Complexity . . . 45
- A.7 Finding Significant Levels . . . 46

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 [Gol17, 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 [GGR98; RS96] is on approximate decision procedures that read small portion of their input (see [Gol17] 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) [EKR04; RVW13]. In such proof systems, *verification should require significantly less queries than testing*, and suitable notions of completeness and soundness should hold.[2] Specifically,

[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).

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* dsIPP*s*?

We stress that the notion of double-efficiency employed here is different of the notion employed in the context of IPs (see [GKR15; RRR21; Gol18]) and also in studies of IPPs that made reference to doubly-efficient IPPs (e.g. [RR20]). 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 [New02] 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]

[3]Consider the set property $\{0^i1^{n-i} : i \in [n]\}$, 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.

Theorem 1.1 (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 4.11 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.

Our protocol builds on an IPP of Goldreich, Gur and Rothblum [GGR18], in which the honest prover reads the entire input. On the other hand, the IPP of [GGR18] 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 [GGR18] (or the tester of [New02], 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 [PRR06] 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 1.1 follows by plugging any tolerant tester for constant-width ROOBPs into our general reduction (which is stated in Theorem 1.2). While, as noted above, Newman's tester for ROOBPs [New02] is not tolerant, we show that that it can be made tolerant. This contribution of the current work is presented in Appendix A.

Our reduction of the prover's startegy to tolerant testing ROOBPs actually gives a *tolerant* dsIPP of ROOBPs, where *Tolerant* IPP*s* 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

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.

[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 2.2).

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 1.2 (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 $\mathsf{Q}_w(\delta, \eta, n)$ and time-complexity $\mathsf{T}_w(\delta, \eta, n)$, 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 $O(n^{1/r(n)} \cdot \mathsf{Q}_w(\delta, \eta)/\delta)$, where $\delta = (\varepsilon_f - \varepsilon_c)/3r(n)$ and $\eta = \delta/O(n^{1/r(n)})$.*
- *The communication complexity is $O(n^{1/r(n)} \cdot r(n) \cdot (\log n)/\delta)$.*
- *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 4.10 for a full and formal statement. As discussed above, Theorem 1.1 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 1.2 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 1.1 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 [Pud84], 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 1.3 (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 3.2 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 [CEG95]).

[6] In contrast, Theorem 1.2 holds also for varying width, but in that case the complexities have a multiplicative facor of $\mathsf{poly}(w)$.

Rothblum, Vadhan and Wigderson [RVW13] 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 [GGR18]. 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 [GLR21].[7] Furthermore, PERM has two different (1-round) IPPs, which were presented in [GLR21] and [GGR18], respectively. We review them next:

1. In the IPP of [GLR21] 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 [GGR18], 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 a honest prover that makes $o(n)$ queries. Interestingly, this "working conjecture" is wrong.

Theorem 1.4 (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 1.4. 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,

[7] See also a direct proof in [Str24, Apdx A].

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 permutaion 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 alsdo 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 1.4.

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., [Gol17, 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 eaches 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 1.5 (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 $\mathsf{polylog}(n)/\varepsilon$*, and the communication complexity is* $O(\log(n)/\varepsilon)$.

Related work. Goldreich and Ron [GR02] showed a $\Omega(\sqrt{n})$ lower bound on the query complexity of testing bipartiteness.[8] A later work of the same authors [GR99] 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 [RVW13] 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 honest prover is not sub-linear: it has to read the entire graph. Our proof of Theorem 1.5 uses the [RVW13] 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 1.5: The sublinear-time honest prover. We first recall the [RVW13] protocol. In the $\mathsf{basic\ test}$, 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 [RVW13], 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 s. 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 [RVW13] 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 1.6, 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 $2/3$.

[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 [Gol17, Sec. 9.3.1]). In particular, the distribution of NO cases is concentrated on expander graphs (see [Gol17, pp. Clm. 9.18.1]). A similar analysis shows that the YES cases are bipartite expanders (w.h.p.)

Claim 1.6 *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 [Gol+21] 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 [Gur+24]). 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 [CG18; HR22; HR23] 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 [RVW13; GGR18], but deviate from it in a significant manner. Loosely speaking, the IPPs of [RVW13; GGR18] rely on partitioning the original property to sub-properties that refer to parts of the original input. However, while the IPPs of [RVW13; GGR18] 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 [RVW13], 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 [GGR18], 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 [Gol17] for an introduction to property testing: the field that studies algorithms of sublinear query complexity for such approximate decision problems.

Definition 2.1 (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 IPP*s 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* IPP*s 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 2.2 *We emphasize that, while in the context of testing, tolerant testing and distance approximation are equivalent [PRR06] (under suitable parameters), this is not the case for* IPP*s.*

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.*

3 ds-IPP for the Hamming Weight Problem

In this section, we present a doubly sub-linear IPP (ds-IPP) 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 $\mathcal{V}$ verifies that $\mathtt{wt}(x) = \sigma$ using query complexity of $O(\frac{1}{\varepsilon})$, the honest prover $\mathcal{P}$ 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 $\mathcal{P}$ computes exact weights for sub-sequences of the input x. Then, in Section 3.2, we present a warm-up towards a ds-IPP: We amend the standard IPP by allowing $\mathcal{P}$ to approximate the Hamming weight of the sub-sequences, and relaxing the accepting conditions of $\mathcal{V}$ in a way that doesn't compromise the completeness and soundness of the protocol. Unfortunately, in this IPP the prover $\mathcal{P}$ still has query complexity of $\Omega(n)$. Lastly, in Section 3.3, we present the actual ds-IPP, which applies the IPP presented in Section 3.2 recursively, thus improving the query complexity of $\mathcal{P}$ without compromising the query complexity of $\mathcal{V}$.

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: $\mathcal{P}$ sends $(\sigma_1, ..., \sigma_k) \leftarrow (\mathtt{wt}(x_1), ..., \mathtt{wt}(x_k))$ to $\mathcal{V}$. Then, $\mathcal{V}$ 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 $\mathcal{V}$ interacts with $\mathcal{P}$, 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 ds-IPP

In this section, we aim to achieve poly-logarithmic query complexity for the honest prover $\mathcal{P}$ without compromising the query complexity of the verifier $\mathcal{V}$: We amend the interaction described in Section

3.1 by allowing $\mathcal{P}$ to *approximate* the Hamming weight of the sub-parts of x. However, at the end of this section, we explain why $\mathcal{P}$ still has query complexity of $\Omega(n)$. In Section 3.3, we show how to solve this issue and present the actual ds-IPP.

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, $\mathcal{P}$ sends $(\sigma_1, ..., \sigma_k)$ to $\mathcal{V}$, 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 $\mathcal{V}$ allows this deviation (and doesn't reject $\mathcal{P}$'s approximations). That is, $\mathcal{V}$ 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 $\mathcal{P}$, 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 $\mathcal{V}$ and $\mathcal{P}$:

- The query complexity of $\mathcal{V}$ 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 $\mathcal{P}$ 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 $\mathcal{V}$ 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 $\mathcal{P}$ 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 $\mathcal{V}$, **and** poly-logarithmic query complexity for $\mathcal{P}$.

3.3 The Actual ds-IPP

In Section 3.2, we aim to achieve poly-logarithmic query complexity for the honest prover $\mathcal{P}$, without compromising the query complexity of the verifier $\mathcal{V}$. Unfortunately, for reasons explained at Section 3.2.2, $\mathcal{P}$ 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 $\mathcal{V}$ 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, $\mathcal{V}$ 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 $\mathcal{V}$ 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. $\mathcal{P}$ 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 $\mathcal{V}$ interacts with $\mathcal{P}$, 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 $\mathcal{P}$ 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 $\mathcal{P}$ also performs weight approximations in the current recursion level, we need to account for both deviations. Details follow.

Protocol 3.1 *ds-IPP 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. $\mathcal{V}$*: If* $r = 0$ *then accept iff* $|\mathtt{wt}(x) - \sigma| \leq \frac{\delta n}{2}$*.*
  3. $\mathcal{P}$*:*
     (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* $\mathcal{V}$*.*
  4. $\mathcal{V}$*:*

(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* $\mathcal{P}$.

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* $\mathcal{V}$ *accepts in all repetitions.*

We show that Protocol 3.1 is a ds-IPP 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 $\mathcal{V}$ interacts with $\mathcal{P}$. 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 $\mathcal{P}$ sends $(\sigma'_1, ..., \sigma'_k)$ to $\mathcal{V}$ 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 $\mathcal{P}$ 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 $\mathcal{V}$ 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 3.1. Let us start by analyzing the query complexity of $\mathcal{V}$ (denoted $Q_{\mathcal{V}}$) and $\mathcal{P}$ (denoted $Q_{\mathcal{P}}$):

- $Q_{\mathcal{V}} = O(\frac{n}{\varepsilon k^{r_0}}) = O(\frac{1}{\varepsilon})$, because we read a sub-sequence of the input of length $\frac{n}{k^{r_0}} = 1$ for $O(\frac{1}{\varepsilon})$ times.
- $Q_{\mathcal{P}}$ is the query complexity of the approximator times the number of calls $\mathcal{P}$ 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 $\mathcal{P}$ invokes the approximator with error probability parameter $\eta = O(\frac{\varepsilon}{k \cdot r_0})$ and deviation parameter $\delta = O(\frac{\varepsilon}{r_0})$.

– $\mathcal{P}$ 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 $Q_{\mathcal{P}} = \tilde{O}(\frac{n^{\frac{1}{r_0}} \cdot r_0^3}{\varepsilon^3})$.

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 $Q_{\mathcal{P}}$ gets, but this causes the round complexity to grow. Specifically, to minimize $Q_{\mathcal{P}}$, we can set $r_0 = \log(n)$, and we get $Q_{\mathcal{P}} = \tilde{O}(\frac{\log^3(n)}{\varepsilon^3})$.

Now, the communication of Protocol 3.1 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 $\mathcal{P}$ 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 $T_{\mathcal{P}}$) is the same as its query complexity (i.e., $T_{\mathcal{P}} = Q_{\mathcal{P}}$), whereas the time complexity of the verifier (denoted $T_{\mathcal{V}}$) 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 3.2 *For every function $r : \mathbb{N} \to \mathbb{N}$, there exists an $r(n)$-round ds-IPP for the Hamming Weight Problem with $Q_{\mathcal{V}} = O(\frac{1}{\varepsilon})$, $Q_{\mathcal{P}} = \tilde{O}(\frac{n^{\frac{1}{r(n)}} \cdot r(n)^3}{\varepsilon^3})$ 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 ds-IPP is $T_{\mathcal{P}} = Q_{\mathcal{P}}$ and $T_{\mathcal{V}} = O(\frac{n^{\frac{1}{r(n)}} \cdot r(n) \cdot log(\frac{r(n)}{\varepsilon})}{\varepsilon})$.*

4 Tolerant ds-IPP for ROOBPs of Constant Width

In this section, we present a tolerant ds-IPP 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 4.1 (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 4.2 *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 4.3 *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 4.4 *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 4.5 (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 $\mathcal{V}$ uses query complexity of $O(\frac{1}{\varepsilon_f - \varepsilon_c})$, the honest prover $\mathcal{P}$ uses poly-logarithmic query complexity, and the following holds:

- If $\overline{\Delta}(B, x) \leq \varepsilon_c$, then when communicating with $\mathcal{P}$, with probability at least $2/3$ the verifier $\mathcal{V}$ accepts x.
- If $\overline{\Delta}(B, x) > \varepsilon_f$, then when communicating with any prover $\mathcal{P}^*$, with probability at least $2/3$ the verifier $\mathcal{V}$ 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 $\mathcal{P}$ partitions the ROOBP to a sequence of k sub-ROOBPs (see Definition 4.6) according to the accepting path associated with an accepting input that minimizes the distance to x in B (see Definition 4.7). Then, in Section 4.4, we present a warm-up towards the tolerant ds-IPP: We amend the standard tolerant IPP by allowing $\mathcal{P}$ to partition the ROOBP according to an approximated path, and relaxing the accepting conditions of $\mathcal{V}$ in a way that doesn't compromise the completeness and soundness of the protocol. Unfortunately, in this IPP the prover $\mathcal{P}$ still has

query complexity of $\Omega(n)$. Lastly, in Section 4.5, we present the actual tolerant ds-IPP, which applies the IPP presented in Section 4.4 recursively, thus improving the query complexity of $\mathcal{P}$ without compromising the query complexity of $\mathcal{V}$.

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 [GGR18], 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 4.6 (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 4.7 ((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 $\mathcal{P}$ finds the (k, π)-decomposition of B constructed according to an accepting input that minimizes the distance to x, checks the distance of x_i from each of the sub-ROOBPs in the decomposition, and sends $\mathcal{V}$ a succinct representation of the decomposition (for example, the sink of every sub-ROOBP), as well as the obtained distances. After $\mathcal{V}$ receives some decomposition $(B_1, ..., B_k)$ and claimed distances $(\hat{\varepsilon_1}, ..., \hat{\varepsilon}_k)$, $\mathcal{V}$ 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, $\mathcal{V}$ 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 $\mathcal{V}$ interacts with $\mathcal{P}$, 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 $\mathcal{P}^*$ 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 $\mathcal{V}$ 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 ds-IPP

In this section, we aim to achieve poly-logarithmic query complexity for the honest prover $\mathcal{P}$ without compromising the query complexity of the verifier $\mathcal{V}$: We amend the interaction described in Section 4.3 by allowing $\mathcal{P}$ 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 $\mathcal{P}$ still has query complexity of $\Omega(n)$. In Section 4.5, we show how to solve this issue and present the actual tolerant ds-IPP.

In Appendix A, 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: $\mathcal{P}$ performs distance approximation (with parameters δ and η we fix later) for every (k, π)-decomposition (constructed according to every accepting path π). Since $\mathcal{P}$ performs distance approximation also for the decomposition of B constructed according to an accepting input that minimizes the distance to x, $\mathcal{P}$ can just choose the the (k, π)-decomposition which yields the minimal (average) distance approximation. That is, $\mathcal{P}$ 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 $\mathcal{P}$ 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 $\mathcal{V}$'s accepting conditions in the interaction, so that $\mathcal{V}$ allows the deviation. That is, after $\mathcal{V}$ 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, $\mathcal{V}$ verifies that $\sum_i \frac{\hat{\varepsilon}_i}{k} \leq \varepsilon_c + \delta$, and after $\mathcal{V}$ chooses $i \in [k]$ uniformly at random, $\mathcal{V}$ 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 $\mathcal{V}$ interacts with $\mathcal{P}$, 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 $\mathcal{P}^*$ 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 $\mathcal{V}$ and $\mathcal{P}$:

- Because $\mathcal{V}$ can verify that the (k, π)-decomposition that $\mathcal{P}$ sends is valid without any query access to the input x, the query complexity of $\mathcal{V}$ is $O(\frac{n}{(\varepsilon_f - \varepsilon_c) \cdot k})$: $\mathcal{V}$ 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 $\mathcal{P}$ 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 $\mathcal{P}$ 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 $\mathcal{V}$ 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 $\mathcal{P}$ 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 $\mathcal{V}$, **and** poly-logarithmic query complexity for $\mathcal{P}$.

4.5 The Actual ds-IPP

In Section 4.4, we aim to achieve poly-logarithmic query complexity for the honest prover $\mathcal{P}$, without compromising the query complexity of the verifier $\mathcal{V}$. Unfortunately, for reasons explained at Section 4.4.2, the prover $\mathcal{P}$ 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 $\mathcal{V}$ 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', $\mathcal{V}$ 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 $\mathcal{V}$ independent of the input length, we set $k = n^{\frac{1}{r}}$.

Now, we observe the following: Assume $\mathcal{V}$ interacts with $\mathcal{P}$, 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 $\mathcal{P}$'s claim regarding the distance of x from B. In the first recursion level, we set $\hat{\varepsilon} = \varepsilon_c$, since $\mathcal{P}$ 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 $\mathcal{P}$ sends the decomposition $(B_1, ..., B_k)$ and the respective approximations $(\hat{\varepsilon}'_1, ..., \hat{\varepsilon}'_k)$. Then, $\mathcal{V}$ 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, $\mathcal{V}$ 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, $\mathcal{P}$ only claims that the average distance doesn't grow too much **with respect to the approximated distance in the previous recursion level**. Since $\mathcal{P}$ 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 $\mathcal{P}$ 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 $\mathcal{V}$ to account for two possible deviations: In the approximation that $\mathcal{P}$ provides for the previous recursion level (i.e. $\hat{\varepsilon}$), and in the approximations that $\mathcal{P}$ performs in the current recursion level (i.e. $(\hat{\varepsilon}_1, ..., \hat{\varepsilon}_k)$).

For soundness, we observe that $\mathcal{V}$ 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. $\mathcal{V}$ 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 4.8 *Tolerant ds-IPP 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. $\mathcal{V}$*: If* $r = 0$ *then accept iff* $\overline{\Delta}(B, x) \leq \hat{\varepsilon} + \frac{\delta}{2}$.
3. $\mathcal{P}$*:*
   (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* $\mathcal{V}$.
4. $\mathcal{V}$*:*
   (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* $\mathcal{P}$.
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* $\mathcal{V}$ *accepts in all repetitions.*

Remark 4.9 $\mathcal{P}$ *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* $\mathcal{P}$ *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,* $\mathcal{P}$ *can choose the decomposition with the shortest* $s \rightsquigarrow t$ *path in* G.

We show that Protocol 4.8 is a ds-IPP 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 $\mathcal{V}$ interacts with $\mathcal{P}$. 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 $\mathcal{P}$ sends $(B'_1, ..., B'_k)$ and $(\hat{\varepsilon}'_1, ..., \hat{\varepsilon}'_k)$ to $\mathcal{V}$ 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 $\mathcal{P}$ 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 $\mathcal{V}$ 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 4.8)

The query and communication complexity of our tolerant ds-IPP 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 $\mathcal{P}$ sends k vertices and k distance approximations to $\mathcal{V}$ 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 $\mathcal{V}$ uses query complexity of $O(\frac{1}{\varepsilon_f-\varepsilon_c})$, and the honest prover $\mathcal{P}$ uses poly-logarithmic query complexity.

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

4.6.1 Query Complexity

Let us analyze the query complexity of $\mathcal{V}$ (denoted $Q_{\mathcal{V}}$) and $\mathcal{P}$ (denoted $Q_{\mathcal{P}}$):

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

- $\mathcal{P}$ uses a distance approximation algorithm $\mathcal{A}$ 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 $Q_{\mathcal{A}}(\delta,\eta,w)$. Then, the query complexity of $Q_{\mathcal{P}}$ is $Q_{\mathcal{A}}(\delta,\eta,w)$ times the number of calls $\mathcal{P}$ makes to $\mathcal{A}$ in all the repetitions. Since $\mathcal{P}$ 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 $Q_{\mathcal{P}} = O(\frac{w^2\cdot n^{\frac{1}{r_0}}\cdot r_0}{\varepsilon_f-\varepsilon_c}\cdot Q_{\mathcal{A}}(\delta,\eta,w))$.

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 $Q_{\mathcal{P}}$ gets, but this causes the round complexity to grow. To minimize $Q_{\mathcal{P}}$, we can set $r_0 = \log(n)$, and we get $Q_{\mathcal{P}} = O(\frac{w^2\cdot\log(n)}{\varepsilon_f-\varepsilon_c}\cdot Q_{\mathcal{A}}(\delta,\eta,w))$.

In Section A.6.1, we show that the query complexity of the distance approximation algorithm presented in Appendix A is $Q_{\mathcal{A}}(\delta,\eta,w) = (\frac{\log(\frac{1}{\eta})\cdot 2^w}{\delta})^{O(w)}$. If $\mathcal{P}$ uses this distance approximation

algorithm, we get $Q_{\mathcal{P}} = \tilde{O}(n^{\frac{1}{r_0}}) \cdot (\frac{r_0 \cdot 2^w}{\varepsilon_f - \varepsilon_c})^{O(w)}$. 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 $T_{\mathcal{V}}$) and the honest prover (denoted $T_{\mathcal{P}}$) is $\Omega(n)$:

- Given a sub-sequence of vertices $(v_0, ..., v_k)$ in B, the verifier $\mathcal{V}$ 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 $\Omega(n)$.
- $\mathcal{P}$ 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 in Appendix A, this also yields time complexity of $\Omega(n)$.

However, we can take an alternative approach in the analysis: Assume both $\mathcal{V}$ and $\mathcal{P}$ 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 $\mathcal{P}$ uses a distance approximation algorithm $\mathcal{A}$ 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 $T_{\mathcal{A}}(\delta, \eta, n, w)$ (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 $\mathcal{V}$ and $\mathcal{P}$ *in the i'th round of the protocol*:

- $\mathcal{V}$ 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.
- $\mathcal{P}$ calls the first procedure 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, $\mathcal{P}$ 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 $O(w^2 \cdot k \cdot (\log(w \cdot n) + T_{\mathcal{A}}(\delta, \eta, \frac{n}{k^i}, w)))$.

Since $\mathcal{V}$ and $\mathcal{P}$ 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:

- $T_{\mathcal{V}} = O(\frac{n^{\frac{1}{r_0}} \cdot r_0 \cdot \log(w \cdot n)}{\varepsilon_f - \varepsilon_c})$
- $T_{\mathcal{P}} = O(\frac{w^2 \cdot n^{\frac{1}{r_0}} \cdot (r_0 \cdot \log(w \cdot n) + \sum_{i=1}^{r_0} T_{\mathcal{A}}(\delta, \eta, n^{\frac{r_0 - i}{r_0}}, w))}{\varepsilon_f - \varepsilon_c})$

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

Theorem 4.10 *For every function $r : \mathbb{N} \to \mathbb{N}$, there exists an $r(n)$-round tolerant ds-IPP for ROOBPs of width w with proximity parameters $\varepsilon_f > \varepsilon_c \geq 0$. In the tolerant ds-IPP, the honest prover $\mathcal{P}$ uses a distance approximation algorithm $\mathcal{A}$ 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 $\mathcal{A}$ is $Q_{\mathcal{A}}(\delta, \eta)$ and the time complexity of $\mathcal{A}$ is $T_{\mathcal{A}}(\delta, \eta, n, w)$. Then, the computational complexity of the tolerant ds-IPP is as follows:*

- ***Query Complexity**: $Q_{\mathcal{V}} = O(\frac{1}{\varepsilon_f - \varepsilon_c})$ and $Q_{\mathcal{P}} = O(\frac{w^2 \cdot n^{\frac{1}{r(n)}} \cdot r(n)}{\varepsilon_f - \varepsilon_c} \cdot Q_{\mathcal{A}}(\delta, \eta))$.*
- ***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 $\mathcal{V}$ and $\mathcal{P}$ have black-box access to procedures that refer to the structure of the input ROOBP, then:*
  - $T_{\mathcal{V}} = O(\frac{n^{\frac{1}{r(n)}} \cdot r(n) \cdot log(w \cdot n)}{\varepsilon_f - \varepsilon_c})$
  - $T_{\mathcal{P}} = O(\frac{w^2 \cdot n^{\frac{1}{r(n)}} \cdot (r(n) \cdot log(w \cdot n) + \sum_{i=1}^{r(n)} T_{\mathcal{A}}(\delta, \eta, n^{\frac{r(n)-i}{r(n)}}, w))}{\varepsilon_f - \varepsilon_c})$

In Section A.6.2, we show that the time complexity of the distance approximation algorithm presented in Appendix A is $T_{\mathcal{A}}(\delta, \eta, n, w) = (\frac{\log(\frac{1}{\eta}) \cdot 2^w}{\delta})^{O(w)} \cdot n$. If $\mathcal{P}$ uses this distance approximation algorithm, we get the following corollary.

Corollary 4.11 *For every function $r : \mathbb{N} \to \mathbb{N}$, there exists an $r(n)$-round tolerant ds-IPP for ROOBPs of width w with $Q_{\mathcal{V}} = O(\frac{1}{\varepsilon_f - \varepsilon_c})$, $Q_{\mathcal{P}} = \tilde{O}(n^{\frac{1}{r(n)}}) \cdot (\frac{r(n) \cdot 2^w}{\varepsilon_f - \varepsilon_c})^{O(w)}$, 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 $\mathcal{V}$ and $\mathcal{P}$ have black-box access to procedures that refer to the structure of the input ROOBP, then $T_{\mathcal{V}} = O(\frac{n^{\frac{1}{r(n)}} \cdot r(n) \cdot log(w \cdot n)}{\varepsilon_f - \varepsilon_c})$ and $T_{\mathcal{P}} = (\frac{r(n) \cdot 2^w \cdot log(n)}{\varepsilon_f - \varepsilon_c})^{O(w)} \cdot n$.*

References

[Pud84] Pavel Pudlák. "A Lower Bound on Complexity of Branching Programs (Extended Abstract)". In: *Mathematical Foundations of Computer Science 1984, Praha, Czechoslovakia, September 3-7, 1984, Proceedings*. Ed. by Michal Chytil and Václav Koubek. Vol. 176. Lecture Notes in Computer Science. Springer, 1984, pp. 480–489. DOI: 10.1007/BFB0030331. URL: https://doi.org/10.1007/BFb0030331.

[CEG95] Ran Canetti, Guy Even, and Oded Goldreich. "Lower Bounds for Sampling Algorithms for Estimating the Average". In: *Inf. Process. Lett.* 53.1 (1995), pp. 17–25. DOI: 10.1016/0020-0190(94)00171-T. URL: https://doi.org/10.1016/0020-0190(94)00171-T.

[RS96] Ronitt Rubinfeld and Madhu Sudan. "Robust Characterizations of Polynomials with Applications to Program Testing". In: *SIAM J. Comput.* 25.2 (1996), pp. 252–271. DOI: 10.1137/S0097539793255151. URL: https://doi.org/10.1137/S0097539793255151.

[GGR98] Oded Goldreich, Shafi Goldwasser, and Dana Ron. "Property Testing and its Connection to Learning and Approximation". In: *J. ACM* 45.4 (1998), pp. 653–750. DOI: 10.1145/285055.285060. URL: https://doi.org/10.1145/285055.285060.

[GR99] Oded Goldreich and Dana Ron. "A Sublinear Bipartiteness Tester for Bounded Degree Graphs". In: *Comb.* 19.3 (1999), pp. 335–373. DOI: 10.1007/S004930050060. URL: https://doi.org/10.1007/s004930050060.

[GR02] Oded Goldreich and Dana Ron. "Property Testing in Bounded Degree Graphs". In: *Algorithmica* 32.2 (2002), pp. 302–343. DOI: 10.1007/S00453-001-0078-7. URL: https://doi.org/10.1007/s00453-001-0078-7.

[New02] Ilan Newman. "Testing Membership in Languages that Have Small Width Branching Programs". In: *SIAM J. Comput.* 31.5 (2002), pp. 1557–1570. DOI: 10.1137/S009753970038211X. URL: https://doi.org/10.1137/S009753970038211X.

[EKR04] Funda Ergün, Ravi Kumar, and Ronitt Rubinfeld. "Fast approximate probabilistically checkable proofs". In: *Inf. Comput.* 189.2 (2004), pp. 135–159. DOI: 10.1016/J.IC.2003.09.005. URL: https://doi.org/10.1016/j.ic.2003.09.005.

[PRR06] Michal Parnas, Dana Ron, and Ronitt Rubinfeld. "Tolerant property testing and distance approximation". In: *J. Comput. Syst. Sci.* 72.6 (2006), pp. 1012–1042. DOI: 10.1016/J.JCSS.2006.03.002. URL: https://doi.org/10.1016/j.jcss.2006.03.002.

[RVW13] Guy N. Rothblum, Salil P. Vadhan, and Avi Wigderson. "Interactive proofs of proximity: delegating computation in sublinear time". In: *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*. Ed. by Dan Boneh, Tim Roughgarden, and Joan Feigenbaum. ACM, 2013, pp. 793–802. DOI: 10.1145/2488608.2488709. URL: https://doi.org/10.1145/2488608.2488709.

[GKR15] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. "Delegating Computation: Interactive Proofs for Muggles". In: *J. ACM* 62.4 (2015), 27:1–27:64. DOI: 10.1145/2699436. URL: https://doi.org/10.1145/2699436.

[Gol17] Oded Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017. ISBN: 978-1-107-19405-2. DOI: 10.1017/9781108135252. URL: http://www.cambridge.org/us/catalogue/catalogue.asp?isbn=9781107194052.

[CG18] Alessandro Chiesa and Tom Gur. "Proofs of Proximity for Distribution Testing". In: *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*. Ed. by Anna R. Karlin. Vol. 94. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018, 53:1–53:14. DOI: 10.4230/LIPICS.ITCS.2018.53. URL: https://doi.org/10.4230/LIPIcs.ITCS.2018.53.

[Gol18] Oded Goldreich. "On Doubly-Efficient Interactive Proof Systems". In: *Found. Trends Theor. Comput. Sci.* 13.3 (2018), pp. 158–246. DOI: 10.1561/0400000084. URL: https://doi.org/10.1561/0400000084.

[GGR18] Oded Goldreich, Tom Gur, and Ron D. Rothblum. "Proofs of proximity for context-free languages and read-once branching programs". In: *Inf. Comput.* 261 (2018), pp. 175–201. DOI: 10.1016/J.IC.2018.02.003. URL: https://doi.org/10.1016/j.ic.2018.02.003.

[RR20] Guy N. Rothblum and Ron D. Rothblum. "Batch Verification and Proofs of Proximity with Polylog Overhead". In: *Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part II*. Ed. by Rafael Pass and Krzysztof Pietrzak. Vol. 12551. Lecture Notes in Computer Science. Springer, 2020, pp. 108–138. DOI: 10.1007/978-3-030-64378-2_5. URL: https://doi.org/10.1007/978-3-030-64378-2%5C_5.

[Gol+21] Shafi Goldwasser et al. "Interactive Proofs for Verifying Machine Learning". In: *12th Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8, 2021, Virtual Conference*. Ed. by James R. Lee. Vol. 185. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, 41:1–41:19. DOI: 10.4230/LIPICS.ITCS.2021.41. URL: https://doi.org/10.4230/LIPIcs.ITCS.2021.41.

[GLR21] Tom Gur, Yang P. Liu, and Ron D. Rothblum. "An Exponential Separation Between MA and AM Proofs of Proximity". In: *Comput. Complex.* 30.2 (2021), p. 12. DOI: 10.1007/S00037-021-00212-3. URL: https://doi.org/10.1007/s00037-021-00212-3.

[RRR21] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. "Constant-Round Interactive Proofs for Delegating Computation". In: *SIAM J. Comput.* 50.3 (2021). DOI: 10.1137/16M1096773. URL: https://doi.org/10.1137/16M1096773.

[HR22] Tal Herman and Guy N. Rothblum. "Verifying the unseen: interactive proofs for label-invariant distribution properties". In: *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*. Ed. by Stefano Leonardi and Anupam Gupta. ACM, 2022, pp. 1208–1219. DOI: 10.1145/3519935.3519987. URL: https://doi.org/10.1145/3519935.3519987.

[HR23] 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*. IEEE, 2023, pp. 743–751. DOI: 10.1109/FOCS57990.2023.00049. URL: https://doi.org/10.1109/FOCS57990.2023.00049.

[Gur+24] Tom Gur et al. "On the Power of Interactive Proofs for Learning". In: *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*. Ed. by Bojan Mohar, Igor Shinkar, and Ryan O'Donnell. ACM, 2024, pp. 1063–1070. DOI: 10.1145/3618260.3649784. URL: https://doi.org/10.1145/3618260.3649784.

[Str24] Hadar Strauss. "Emulating Computationally Sound Public-Coin IPPs in the Pre-Coordinated Model". In: *Electronic Colloquium on Computational Complexity (ECCC)* TR24-131 (2024). URL: https://eccc.weizmann.ac.il/report/2024/131/.

A δ-Distance-Approximation Algorithm for an ROOBP of Constant Width

A.1 Introduction

In the next sections we present and analyze a distance-approximation algorithm for ROOBPs of constant width. We refer the reader to some standard definitions related to ROOBPs in Section 4.1. Here, we define the notion of a distance-approximation algorithm for ROOBPs.

Definition A.1 (δ-Distance-Approximation Algorithm for B) *A* ***δ-distance approximation algorithm for*** *B is an algorithm that is given "free" access to the ROOBP B, query access to the input $x \in \{0,1\}^n$, a deviation parameter $\delta > 0$ and error probability η, and outputs $\hat{\varepsilon}$ such that $\hat{\varepsilon} \in (\overline{\Delta}(x, B) \pm \delta)$ with probability at least $1 - \eta$.*

Note: Our goal is to find such an algorithm A while minimizing its query complexity. We show that our algorithm, assuming the width of the ROOBP and the error probability is constant, has query complexity $\text{poly}(1/\delta)$. We defer the complexity discussion to Section A.6.

The following is our high-level plan for doing so:

1. We present a δ-distance-approximation algorithm for the case $w = 1$. This is our base case, so we can assume we have a δ-distance-approximation algorithm for any ROOBP of width at most $w - 1$, and provide an approximation for width at most w.
2. We show that if B is "w-sparse" (see Definition A.3), then we can reduce the problem of distance approximation for it, to approximating the distance of not-too-many simplified sub-ROOBPs (see Definition A.7) of B, each of width at most $w - 1$.
3. Building on 2, we show that if B is "locally w-sparse" (see Definition A.19), then we can reduce the problem of distance approximation for it, to approximating the distance of not-too-many sparse simplified sub-ROOBPs of it.
4. We construct an "approximating locally w-sparse" decomposition of B to sub-ROOBPs, $\{B_1, ..., B_h\}$. The decomposition has the following properties:
   (a) Each B_i is either "locally w-sparse", or relatively small.
   (b) Approximating the distance from B can be reduced to approximating the distance from each sub-ROOBP in a not-too-big subset of $\{B_1, ..., B_h\}$.
5. Lastly, we show a δ-distance-approximation algorithm for B using the above decomposition.

A.2 δ-Distance-Approximation Algorithm for a ROOBP of width 1

In the case B is of width 1, we can assume without loss of generality that B has $n' \leq n$ levels, such that each level i "verifies" that x_i is equal to some predefined $\sigma_i \in \{0,1\}$ (see Figure A.2). This is true for every level with a single outgoing edge, and for a level with two outgoing edges, we can simply remove the level, because it doesn't affect our computation. Using Remark 4.4, we can assume $n' = n$.

In other words, B verifies that an input $x \in \{0,1\}^n$ equals some predefined string $\sigma \in \{0,1\}^n$. For this property, we can perform δ-distance-approximation with error probability η by choosing

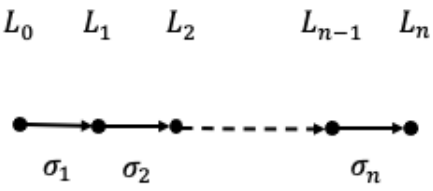

Figure 1: An ROOBP of width 1

uniformly at random $O(\frac{\log(\frac{1}{\eta})}{\delta^2})$ indices of x, and outputting the (normalized) number of indices i for which $x_i = \sigma_i$. Using Chernoff Bound, we can show that the procedure δ-approximates, with probability at least $1 - \eta$, the distance of x to σ.

A.3 δ-Distance-Approximation Algorithm for a w-sparse ROOBP

As mentioned in the introduction, we want to reduce the distance approximation from a w-sparse ROOBP B to approximating the distance from narrower (of width at most $w-1$) sub-ROOBPs of B. For that sake, we do the following:

- Define some notions such as a w-sparse ROOBP and a sub-ROOBP.
- Show that in a w-sparse B, there are sub-ROOBPs that yield the aforementioned reduction, and show how these sub-ROOBPs can be efficiently found.
- Assume we have a δ'-Distance Approximation Algorithm (denoted $A^{w-1}_{\delta',\eta'}$) for $\delta' > 0$ and B' of length n' and width at most $w-1$ with error probability η'. Using $A^{w-1}_{\delta',\eta'}$ and our reduction, we construct a δ-Distance Approximation Algorithm for $\delta > 0$ and a w-sparse B of length n and width at most w with error probability η.

A.3.1 Definitions

We start with the definitions related to sparsity:

Definition A.2 (w-connectivity) *For $l \neq l' \in \{0, ..., n\}$, a vertex $v \in L_l$ is w-**connected** to $L_{l'}$ if there are w vertices in $L_{l'}$, and v is connected (via a directed path) to each of them. If $l > l'$, then v is **backwards-w-connected** to $L_{l'}$, and if $l < l'$, then v is **forwards-w-connected** to $L_{l'}$. The levels L_l and $L_{l'}$ are w-**connected** if there are w vertices in L_l, and each of them is w-connected to $L_{l'}$. In other words, each of the w vertices of L_l is connected to each of the w vertices of $L_{l'}$.*

Definition A.3 (w-sparse ROOBP) *B is considered w-**sparse** if for **every** $l \neq l' \in \{0, ..., n\}$, L_l and $L_{l'}$ are **not** w-connected .*

We continue by defining the notion of a sub-ROOBP:

Definition A.4 (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 $x[i+1,j] = x_{i+1} \cdot x_{i+2} \cdot \cdots \cdot x_j$ is the input for $B[u,v]$.*

Remark A.5 *For simplicity, we sometimes refer to the input of $B[u,v]$ as x, since the actual input to $B[u,v]$ is implied by the levels of u and v.*

In the algorithm that we present in this section, we modify sub-ROOBPs of B in order to make them narrower (of width at most $w-1$). The modification consists of deleting vertices that don't affect the function that the sub-ROOBP computes.

Definition A.6 (Redundant Vertices) *A vertex v in B is **redundant** if it is not connected (via a directed path) to either the source (i.e. s) or the sink (i.e. t) of B.*

Definition A.7 (Simplified ROOBP) *If we remove all redundant vertices from B, then we get B', which we call the **simplified** version of B. We call this procedure **simplification**, denoted by $B' \leftarrow Simplify(B)$.*

Since our simplification process relies only on deleting redundant vertices, we know that an ROOBP and its simplified version compute the same function.

Now, we assume B is w-sparse and simplified, and that after the simplification process it is still of width w; that is, there still exists $l \in \{0,...,n\}$ such that there are w (non-redundant) vertices in L_l. Therefore, we cannot apply $A^{w-1}_{\delta',\eta'}$ on B. However, we show there exist pairs of connected vertices (u,v) that "dissect" B to sub-ROOBPs such that $B[s,u]$ and $B[v,t]$ are narrower, and thus applicable for $A^{w-1}_{\delta',\eta'}$ (we select u and v to be relatively close and neglect the effect of $B[u,v]$).

Definition A.8 (Narrowing Pair) *Let (u,v) be a pair of vertices in B such that u is before v. Then (u,v) is considered a **narrowing pair** if after simplification, $B[s,u]$ and $B[v,t]$ are of width at most $w-1$.*

However, this is not enough for us, since the distance of x from $B[s,u]$ and $B[v,t]$ might not reflect the distance of x from B. Hence, we want to find pairs that **do** reflect the aforementioned distance.

Definition A.9 (δ-Approximating Pair) *Let (u,v) be a pair of vertices in B such that u is before v. Then (u,v) is considered a δ-**approximating pair** for B if performing distance approximation for B is reducible to performing distance approximation for $B[s,u]$ and $B[v,t]$. That is:*

$$\frac{\Delta(x,B[s,u])+\Delta(x,B[v,t])}{n} \in (\varepsilon \pm \delta)$$

In Section A.3.2 we show how we can achieve this by requiring that u an v are relatively close and reside on the $s \rightsquigarrow t$ path associated with the accepting input that minimizes the distance to the input x.

Note: We will allow $u = s$ (resp., $v = t$), and in this case $B[s,u]$ (resp., $B[v,t]$) is of length 0. This is not problematic for us, because this implies that the sub-ROOBP is of width 1 and always accepts its input (the empty string). Thus, its distance approximation is always $\hat{\varepsilon} = 0$.

If we find a pair in B that is both narrowing and δ-approximating, then we can perform distance approximation for B: Apply $A^{w-1}_{\delta',\eta'}$ on $B[s,u]$ and $B[v,t]$, and output the value specified in Definition A.9.

Definition A.10 (δ-Good Pair) *Let (u,v) be a pair of vertices in B such that u is before v. Then (u,v) is considered a δ-**good pair** for B if it is both narrowing and δ-approximating.*

In Section A.3.2 we show how to find a δ-good pair in a simplified w-sparse ROOBP, and in Section A.3.3 we show how to find one **efficiently**.

A.3.2 Finding of a δ-Good Pair

We first describe a method to find narrowing pairs in a simplified, w-sparse B. We then describe a method to find δ-approximating pairs in any ROOBP. Then, we show how to combine these methods to find at least one pair that is both narrowing and δ-approximating (i.e. a δ-good pair).

Finding Narrowing Pairs We first observe the following: For any ROOBP B of width $w > 1$, there is a trivial narrowing pair: (s, t), since $B[s, s]$ and $B[t, t]$ are both of width 1. However, we show that in a simplified, w-sparse B, we can also find non-trivial narrowing pairs. Moreover, we find narrowing pairs that are relatively close (we need to show there exist a pair that is also δ-approximating).

We continue with showing the following two claims, which state that a sub-ROOBP of B is narrower then B, if some assumption regarding its start or end vertex holds.

Claim A.11 *Let $v \in L_l$ be a vertex in an ROOBP B. If v is not w-connected to any level in $B[s, v]$ (resp., $B[v, t]$), then the simplified $B[s, v]$ (resp., $B[v, t]$) is of width at most $w - 1$.*

Proof: If v is not w-connected to some $L_{l'}$ in $B[s, v]$ (resp., $B[v, t]$), then either there's a redundant vertex in $L_{l'}$, or $L_{l'}$ is already of width at most $w-1$. Therefore, after simplification, $L_{l'}$ is of width at most $w - 1$. ■

Claim A.12 *Let $v \in L_l$ be a vertex in a w-sparse B. If v is backwards-w-connected to **some** level $L_{l'}$, then the simplified $B[v, t]$ is of width at most $w - 1$.*

Proof: We claim that v isn't w-connected to any level in $B[v, t]$: If there is a level $L_{l''}$ in $B[v, t]$ (i.e. $l'' > l$) such that v is w-connected to $L_{l''}$, then it means that $L_{l'}$ and $L_{l''}$ are w-connected (see Figure 2), which contradicts the hypothesis that B is w-sparse. Therefore, by Claim A.11, the simplified $B[v, t]$ is of width at most $w - 1$. ■

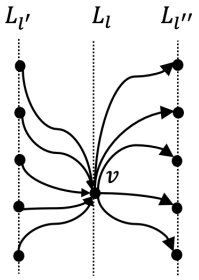

Figure 2: $L_{l'}$ and $L_{l''}$ are w-connected via v

We now define the set P as the set of all vertices that are backwards-w-connected to some level. We use P to find non-trivial narrowing pairs in a w-sparse, simplified B as follows.

Claim A.13 *Let B be a w-sparse ROOBP of width w. Let (u, v) be a pair of vertices in B such that u is before v. If $u \notin P$ and $v \in P$, then (u, v) is a narrowing pair in B.*

Proof: Since $u \notin P$, u is not backwards-w-connected to any level, thus not w-connected to any L_l in $B[s, u]$. By Claim A.11, the simplified $B[s, u]$ is of width at most $w - 1$. Since $v \in P$, v is

backwards-w-connected to some level. By Claim A.12, the simplified $B[v,t]$ is of width at most $w-1$. ■

Now, if we assume B is simplified, then Claim A.13 is consistent with (s,t) being a narrowing pair : $s \notin P$, since there is no level before s, and $t \in P$, since B is simplified and of width w, so t must be backwards-w-connected to some level in B. However, using Claim A.13 and the following claim, we can find non-trivial narrowing pairs in B.

Claim A.14 *Let B be an ROOBP of width w. Let $v \in L_l$ such that $v \in P$. Then, for any $v' \in L_{l'}$ that is backwards-connected to v, we have that $v' \in P$.*

Proof: Let $v' \in L_{l'}$ that is backwards-connected to v. Since $v \in P$, we know v is backwards-w-connected to some level $L_{l''}$. Let $v'' \in L_{l''}$. Then, v'' is connected to v, so there exists a $v'' \rightsquigarrow v \rightsquigarrow v'$ path in B. Therefore, v' is connected to each of the w vertices in $L_{l''}$ (see Figure 3), and so $v' \in P$. ■

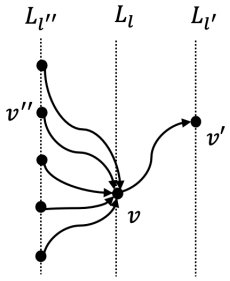

Figure 3: v' is w-connected to $L_{l''}$

We can now describe our method for finding narrowing pairs in a simplified, w-sparse B: Since $t \in P$, we know that P is not empty. So let $v^* \in L_{l^*}$ be a vertex in P such that l^* is minimal (i.e. there is no vertex $u \in L_{l''}$ in P such that $l'' < l^*$). Then, for any u before v^*, and for any v that is backwards-connected to v^*, we have that (u,v) is a narrowing pair. Note that by the minimality of l^* we know that $u \notin P$ and by Claim A.14 we know that $v \in P$. Hence, by Claim A.13, (u,v) is a narrowing pair.

Note that at least one of the narrowing pairs in B found by the method above are in adjacent levels, thus are not trivial. We use a similar observation later in this section to find narrowing pairs that are relatively close, and thus can be δ-approximating.

Finding δ-Approximating Pairs We start with the following observation: Let $x' \in \{0,1\}^n$ be an accepting input that minimizes the distance to the input x; that is, $\Delta(x',B) = 0$ and $\Delta(x,B) = \Delta(x,x') = \varepsilon n$. Let $(u_0 = s, u_1, ...u_{n-1}, u_n = t)$ be the $s \rightsquigarrow t$ path associated with x'. Since the distance of x to B is defined by the distance of x to x', it holds that for any $0 \leq i < j \leq n$:

$$\Delta(x,B) = \Delta(x,B[s,u_i]) + \Delta(x,B[u_i,u_j]) + \Delta(x,B[u_j,t])$$

Therefore, any pair (u_i,u_j) along the path such that $0 \leq j-i \leq \delta n$ is a δ-approximating pair: We can just omit the distance that $B[u_i,u_j]$ contributes, because it is at most δ (in relative distance).

Finding δ-Good Pairs We now combine the two methods shown in Sections A.3.2 to find δ-good pairs in a simplified, w-sparse ROOBP B.

As shown in Section A.3.2, any pair (u_i, u_j) along the path associated with x' (an accepting input that minimizes the distance to the input x) such that $0 \leq j - i \leq \delta n$ is a δ-approximating pair. We also know that $s = u_0 \notin P$ and $t = u_n \in P$. Thus we can look at the minimal $j^* > 0$ such that $u_{j^*} \in P$, and we get:

- For any $i < j^*$, by the minimality of j^*, we know that $u_i \notin P$.
- For any $j \geq j^*$, u_j is backwards-connected to u_{j^*}, since they are part of the path associated with x'. By Claim A.14, $u_j \in P$.

Then, by Claim A.13, the simplified $B[s, u_i]$ and $B[u_j, t]$ are both of width at most $w-1$. Restricting i and j such that $j \leq i + \delta n$, which means that $(j - \delta n) \leq i < j^* \leq j$, will give us narrowing, δ-approximating pairs (which are δ-good pairs).

A.3.3 The Actual Algorithm

In Section A.3.2, we show how to find a δ-good pair in a w-sparse, simplified B. Recall that if we find a δ-good pair in B, we can perform distance approximation for B: We apply $A^{w-1}_{\delta',\eta'}$ on $B[s, u]$ and $B[v, t]$, and output the value specified in Definition A.9. However, since we don't know x' (since our algorithm cannot look at the entire input x), we can't use the method described in Section A.3.2 for finding a δ-good pair. In this section, we show how we can **efficiently** find a δ-good pair, and use it to present our δ-approximation algorithm for a w-sparse ROOBP.

We can try to overcome the above problem by using the following observation: In every $s \rightsquigarrow t$ path, associated with any x'', there are narrowing pairs (u_i, u_j) that reside on this path, such that $(j - \delta n) \leq i < j$. The method to finding them is similar to the one described in in Section A.3.2. For every such pair, we know that $\Delta(x, x'') \geq \Delta(x, x')$, and therefore:

$$\Delta(x, B) \leq \Delta(x, B[s, u_i]) + \Delta(x, B[u_i, u_j]) + \Delta(x, B[u_j, t])$$

Thus, we can apply the reduction for such a pair in every possible path from s to t, and since we also try a pair which belongs to x', we can just choose the minimal value obtained by the reduction. But still, we can't use the above method, since there could be too many such paths.

Our solution is to restrict the vertex pairs to be in small number of special levels that are of distance δn from each other. Instead of choosing a narrowing pair for each $s \rightsquigarrow t$ path, we choose **every possible** pair in these levels such that the pair (u, v) is:

- Narrowing, so simplifying $B[s, u]$ and $B[v, t]$ yields width $w - 1$.
- Connected, so (u, v) is part of some $s \rightsquigarrow t$ path.
- In adjacent special levels, so (u, v) will be at distance at most δn.

Since all paths go through all levels, the above guarantees that we choose at least one pair for each path. Specifically, we show that we choose at least one δ-good pair in the path associated with x'. Since the number of special levels is small (and w is constant), we can afford to perform the reduction to all possible pairs, and output the minimum value.

Indeed, we might "pay" in distance per the deviation that $A^{w-1}_{\delta',\eta'}$ may have (in the distance approximation of x from $B[s, u]$ and $B[v, t]$). Therefore, we set a different deviation parameter (i.e.

δ') for the recursive calls than our input parameter, δ. In addition, we want that with probability at least $1-\eta$, **all** the calls to $A^{w-1}_{\delta',\eta'}$ are successful (i.e. the distance approximation for each sub-ROOBP provided by $A^{w-1}_{\delta',\eta'}$ does not deviate by more than δ'). Therefore, we set a different error probability parameter (i.e. η') for the recursive calls than our input error probability parameter, η. Details follow.

Algorithm A.15 *δ-Distance Approximation Algorithm for w-sparse ROOBP*

- ***Input (Query Access)**: $x \in \{0,1\}^n$*
- ***Massive Parameter**: A w-sparse, simplified B of length n, width at most w, source vertex s and sink vertex t*
- ***Deviation Parameter**: $\delta > 0$*
- ***Error Probability**: $\eta > 0$*

1. *Define $\delta' = \frac{\delta}{2}$ and $\eta' = \frac{\eta \cdot \delta'}{w^2} = \frac{\eta \cdot \delta}{2w^2}$. Add (u,v) to a list of pairs, S, if the following holds:*
   - *There exists $i \in [\frac{1}{\delta'}]$ such that $u \in L_{(i-1)\delta' n}$ and $v \in L_{i\delta' n}$.*
   - *(u,v) is a narrowing pair (determining whether a pair is narrowing can be done using Algorithm A.37, see more details in Section A.7).*
   - *u is connected to v.*
2. *For every $(u,v) \in S$, do the following:*
   (a) *Approximate the distance of x from $B[s,u]$, denoted $\hat{\varepsilon}_1$, by applying $A^{w-1}_{\delta',\eta'}$ on the input x, with the the simplified $B[s,u]$, deviation parameter δ' and error probability η'.*
   (b) *Approximate the distance of x from $B[v,t]$, denoted $\hat{\varepsilon}_2$, by applying $A^{w-1}_{\delta',\eta'}$ on the input x, with the the simplified $B[v,t]$, deviation parameter δ' and error probability η'.*
   (c) *Calculate the weighted distance approximation:*
      i. *Define $l_1 = (i-1)\delta' n$ as the length of B_1, and $l_2 = n - i\delta' n$ as the length of B_2.*
      ii. *$\hat{\varepsilon}_{(u,v)} = \frac{\hat{\varepsilon}_1 l_1 + \hat{\varepsilon}_2 l_2}{n}$.*
3. ***Return** $\min\{\hat{\varepsilon}_{(u,v)} : (u,v) \in S\}$*

Remark A.16 *The conditions of Step 1 depend only on the structure of B and therefore can be tested without any query access to the input x. Thus, the query complexity of Algorithm A.15 depends only on the query complexity of $A^{w-1}_{\delta',\eta'}$ and on the number of recursive calls we make to $A^{w-1}_{\delta',\eta'}$, which is bounded by $O(\frac{w}{\delta^2})$.*

We show the algorithm above is a δ-approximation algorithm for a w-sparse and simplified ROOBP:

First, since every pair $(u,v) \in S$ is narrowing, we can indeed apply $A^{w-1}_{\delta',\eta'}$ on $B[s,u]$ and $B[v,t]$. Then, we observe that we only use randomization via the recursive calls to $A^{w-1}_{\delta',\eta'}$. Since we call $A^{w-1}_{\delta',\eta'}$ for at most $\frac{w^2}{\delta'}$ times, and since $\eta' = \frac{\delta' \cdot \eta}{w^2}$, we get that with probability at least $1-\eta$, all the calls to $A^{w-1}_{\delta',\eta'}$ are successful (i.e. provide distance approximation for each sub-ROOBP with deviation of at most δ'). Therefore, all we have left is to show that, assuming all the calls to $A^{w-1}_{\delta',\eta'}$ are successful, the approximation that Algorithm $A.15$ provides deviates by at most δ.

We claim that one of the pairs in S is a δ'-good pair: Let $(u_0 = s, u_1, ...u_{n-1}, u_n = t)$ be the $s \rightsquigarrow t$ path associated with x' (an accepting input that minimizes the distance to the input x). Recall that in Section A.3.2, we showed the following:

- There exists minimal $j^* > 0$ such that $u_{j^*} \in P$ (on the path associated with x').
- Any (u_i, u_j) such that $i < j^* \leq j$ and $(j - \delta' n) \leq i < j$ is a δ'-good pair.

Now, let $i \in \{0, ..., n\}$ be the minimal such that $j^* \geq i\delta' n$. If $j^* = i\delta' n$, then the pair $(u_{(i-1)\delta' n}, u_{i\delta' n})$ is a δ'-good pair in S. Else, the pair $(u_{i\delta n}, u_{(i+1)\delta' n})$ is a δ'-good pair in S. Then, for this pair, we have:

$$\Delta(B, x) = \Delta(B[s, u], x) + \Delta(B[u, v], x) + \Delta(B[v, t], x)$$

Since any other pair $(u, v) \in S$ is connected, there exists an $s \rightsquigarrow u \rightsquigarrow v \rightsquigarrow t$ path in B. Therefore:

$$\Delta(B, x) \leq \Delta(B[s, u], x) + \Delta(B[u, v], x) + \Delta(B[v, t], x)$$

Thus, by taking the minimal approximated distance obtained in Step 2, our approximation deviates from ε only by the following additive factors:

1. The deviation in the approximation of $A^{w-1}_{\delta',\eta}$. By the correctness of $A^{w-1}_{\delta',\eta}$ this deviation is at most δ'.
2. The deviation due the fact that we don't approximate $\overline{\Delta}(B[u, v], x)$. Since the distance between u and v is $\delta' n$, the deviation is bounded by δ'.

Hence, our approximation deviates by a factor of at most $2\delta' = \delta$. We conclude with the following theorem, that summarizes what we achieved in this section.

Theorem A.17 *δ-distance-approximation for w-sparse ROOBPs with error probability η reduces to $O(\frac{w^2}{\delta})$ calls to $\frac{\delta}{2}$-distance-approximation for ROOBPs of width $w - 1$ with error probability $O(\frac{\delta}{2w^2} \cdot \eta)$.*

A.4 δ-Distance-Approximation Algorithm for a (α, β, w)-locally-sparse ROOBP

In this section, we present a relaxation of the sparsity requirement (see Definition A.19). We show that we can still perform distance approximation for an ROOBP that satisfies the relaxed sparsity requirement, by decomposing the ROOBP to sub-ROOBPs, and performing distance approximation for each sub-ROOBP using Algorithm A.15 as a black box.

In Section A.5, we decompose any ROOBP to sub-ROOBPs that satisfy the relaxed requirement, and perform distance approximation for each sub-ROOBP using the algorithm we present in this section.

We start with definitions related to the relaxed sparsity requirement.

Definition A.18 (d-Close Levels) *Let B be an ROOBP. We say that L_l and $L_{l'}$ are d-**close** if $|l - l'| < d$.*

Our relaxed notion of sparsity is to require sparsity only locally. In addition, we allow some slackness in the first levels of the ROOBP. We need this slackness in Section A.5.1, where we show that we can decompose any ROOBP to sub-ROOBPs that satisfy the relaxed sparsity requirement.

Definition A.19 ((α, β, w)-locally-sparse ROOBP) *Let B be an ROOBP of width w and length n. Then B is considered (α, β, w)-**locally-sparse** if **every** pair of αn-close levels $(L_l, L_{l'})$ with $l, l' > \beta n$ is **not** w-connected.*

In the δ-approximation algorithm that we present in this section, we require that $\alpha = \Omega(\delta^2)$, to make sure the locality (defined by α) is not too granular (see Section A.5.1). In addition, we require that $\beta \leq \frac{\delta}{2}$, to make sure we can neglect the effect of the first βn levels of B (in case they don't follow the local-sparsity requirement).

We continue with defining notions related to decomposing an ROOBP. We first present a general notion of a decomposition.

Definition A.20 (Decomposition and Dissection Levels) *Let B be an ROOBP. A **decomposition** of B is a set of sub-ROOBPs of B, denoted $\mathcal{B} = (B_1, ..., B_h)$, such that:*

- *The source of B_1 is s, and the sink of B_h is t.*
- *For every $i \in [h]$, the source and sink of B_i are connected.*
- *For every $i \in [h-1]$, the sink of B_i and the source of B_{i+1} are in the same level. That is, the sub-ROOBPs in the set are consecutive.*

*For every $i \in [h]$, the level of the sink of B_i is called the **dissection level** of B_i.*

We continue by defining a special notion of a decomposition.

Definition A.21 (Connected Decomposition) *We say that a decomposition $\mathcal{B} = (B_1, ..., B_h)$ is **connected** if for every $i \in [h-1]$ the sink of B_i equals the source of B_{i+1}. That is, the sources and sinks of all B_i's reside on an $s \rightsquigarrow t$ path in B.*

In this section, all decompositions we use are connected. This is because we use the decompositions to perform distance approximation (i.e. we perform distance approximation to the ROOBP by performing distance approximation to the sub-ROOBPs in the decomposition), and we can ensure we don't lose distance by using a connected decomposition. In contrast, in Algorithm A.33, we use a decomposition that isn't necessarily connected. In Section A.5.2, we show how to avoid the condition of having the decomposition connected by using the fact that every dissection level (but the last) is backwards-w-connected to a relatively close level. We show that this implies that the decomposition is not far in distance from some connected decomposition (see Claim A.31 for more details).

Now, we observe that if B is (α, β, w)-locally-sparse, we can construct a decomposition of B, denoted $\mathcal{B}' = (B'_1, ..., B'_h)$, with $h = O(\frac{1}{\alpha}) = O(\frac{1}{\delta^2})$, such that:

- B'_1 contains the first βn levels of B. That is, the dissection level of B'_1 is $L_{\beta n}$.
- For each $i \in \{2, ..., h\}$, B'_i is of length αn. That is, the dissection level of B'_i is $L_{\beta n+(i-1)\alpha n}$.

Then, for every $i \in \{2, ..., h\}$, B'_i is w-sparse regardless of the specific source and sink we choose. This is because every level of B'_i is after $L_{\beta n}$, and every pair of levels in B'_i is αn-close (since B'_i is of length αn). Thus, for an (α, β, w)-locally-sparse B, every pair of levels in B'_i is not w-connected.

Now, we want to define the source and sink of every $B'_i \in \mathcal{B}'$ such that performing distance approximation to B is reducible to performing distance approximation for every $B_i \in \mathcal{B}'$. To achieve this, we can define the source and sink of every B'_i according to the $s \rightsquigarrow t$ path associated with x', an accepting input that minimizes the distance to the input x. That is, let $(u_0 = s, u_1, ...u_{n-1}, u_n = t)$ be the $s \rightsquigarrow t$ path associated with x'. We define $B'_1 = B[s, u_{\beta n}]$, and for every $i \in \{2, ..., h\}$, we define $B'_i = B[u_{\beta n+(i-2)\alpha n}, u_{\beta n+(i-1)\alpha n}]$. Since the distance of x to B is defined by the distance of x to x', we get that:

$$\sum_{i\in[h]} \Delta(x, B'_i) = \Delta(x, B)$$

Therefore, if we assume that $\beta \leq \frac{\delta}{2}$ and $\alpha \geq \text{poly}(\delta)$, we can perform distance approximation for B: For every $i \in \{2, ..., h\}$, we perform distance approximation for the simplified B'_i using Algorithm A.15, and we output the average of all distance approximations. Since $\beta \leq \frac{\delta}{2}$, we can afford to neglect the effect of B'_1, and since $h = O(\frac{1}{\alpha}) = O(\frac{1}{\delta^2})$, we can afford to perform distance approximation for all the other sub-ROOBPs.

We still face a similar issue we faced in Section A.3.3: We cannot construct $\mathcal{B}'$ since we don't know x'.

We can overcome this problem by using the following observation: For every accepting input x'', we can construct a new connected decomposition by changing the source and sink of every B'_i according to the accepting path associated with x'', and get a new decomposition, $(B''_1, ..., B''_h)$. Then, since we don't change the decomposition's dissection levels, every B''_i is w-sparse. Since $\Delta(x, x'') \geq \Delta(x, x')$, we get that:

$$\sum_{i\in[h]} \Delta(x, B''_i) \geq \Delta(x, B)$$

Thus, we can perform distance approximation for every connected decomposition (with the same dissection levels) constructed according to every accepting input. Since we also try a connected decomposition constructed according to x', we can just choose the minimal value obtained by the reduction. Now, even though there could be many such paths, the number of sub-ROOBPs in all possible connected decompositions (with the same dissection levels) is upper-bounded by w^2h: For every dissection level, there are at most w^2 possible source-sink pairs for the sub-ROOBP with this dissection level.

Indeed, we might "pay" in distance per the deviation that A.15 may have. Therefore, we set a different deviation parameter (i.e. δ') for the recursive calls than our input parameter, δ. In addition, we want that with probability at least $1-\eta$, **all** the calls to Algorithm A.15 are successful (i.e. the distance approximation for each sub-ROOBP provided by Algorithm A.15 does not deviate by more than δ'). Therefore, we set a different error probability parameter (i.e. η') for the recursive calls than our input error probability parameter, η. Details follow.

Algorithm A.22 *δ-Distance Approximation Algorithm for (α, β, w)-locally-sparse ROOBP*

- ***Input (Query Access)**: $x \in \{0,1\}^n$*
- ***Massive Parameter**: An (α, β, w)-locally-sparse B of length n, width at most w, source vertex s and sink vertex t.*
- ***Deviation Parameter**: $\delta > 0$*
- ***Error Probability**: $\eta > 0$*
- ***Additional Parameters**: α and β such that $\alpha = \Omega(\delta^2)$ and $\beta \leq \frac{\delta}{2}$.*

1. *Define $\delta' = \frac{\delta}{2}$ and the dissection levels $(L_{\beta n}, L_{\beta n+\alpha n}, L_{\beta n+2\alpha n}, ..., L_n) = (L_{r_1}, ..., L_{r_h})$.*
2. *Add $B[u, v]$ to a list of sub-ROOBPs, R, if the following holds:*
    - *There exists $i \in \{2, ..., h\}$ such that $u \in L_{r_{i-1}}$ and $v \in L_{r_i}$.*

- *u is connected to v.*

3. *Define $\eta' = \frac{\eta \cdot \alpha}{w^2}$, and for every $B[u,v] \in R$, approximate the distance of x from $B[u,v]$ by applying Algorithm A.15 on the input x, with the simplified $B[u,v]$, deviation parameter δ' and error probability η'.*

4. *For every connected decomposition $(B_1, ..., B_h)$ such that $B_2, ..., B_h \in R$, let $\sum_{i \geq 2} \hat{\varepsilon}_i$ be the sum of the distance approximations for $(B_2, ..., B_h)$ obtained in the previous step.* ***Return*** *the minimum of all such sums.*

Remark A.23 *The conditions of Step 2 depend only on the structure of B and therefore can be tested without any query access to the input x. Thus, the query complexity of Algorithm A.22 depends only on the query complexity of Algorithm A.15 and on the number of recursive calls we make to Algorithm A.15, which is bounded by $w^2(h-1) = \frac{w^2}{\alpha} = O(\frac{w^2}{\delta^2})$.*

Remark A.24 *We can efficiently calculate the minimum sum in Step 4 as follows: We construct a graph $G = (V, E)$ with V being all the vertices of B in the levels $(L_0, L_{r_1}, ..., L_{r_h})$, and $u, v \in V$ are connected in G with edge-weight $\hat{\varepsilon}_{u,v}$ if we perform distance approximation for $B[u,v]$ in Step 3 and it results in $\hat{\varepsilon}_{u,v}$. In addition, s is connected to every $v \in L_{r_1}$ with edge-weight 0. Then, every set $B_2, ..., B_h \in R$ considered in Step 4 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 $B_2, ..., B_h$. Thus, we can output the weight of the shortest $s \rightsquigarrow t$ path in G.*

We now show the algorithm above is a δ-approximation algorithm for a (α, β, w)-locally-sparse ROOBP:

We observe that we only use randomization via the recursive calls to Algorithm A.15. Since we call Algorithm A.15 for at most $\frac{w^2}{\alpha}$ times, and since $\eta' = \frac{\eta \cdot \alpha}{w^2}$, we get that with probability at least $1 - \eta$, all the calls to Algorithm A.15 are successful (i.e. provide distance approximation with deviation of at most δ' for each sub-ROOBP). Therefore, all we have left is to show that, assuming all the calls to Algorithm A.15 are successful, the approximation that Algorithm $A.22$ provides deviates by at most δ.

Since in Step 4 we consider all possible connected decompositions (with the dissection levels $(L_{r_1}, ..., L_{r_h})$), one of the sums belong to the connected decomposition constructed according to x' (an accepting input which minimizes the distance to the input x). Thus, by taking the minimum sum in Step 4, our approximation only deviates from ε by the following additive factors:

- The deviation in the approximation of Algorithm A.15. By the correctness of Algorithm A.15, this deviation is at most δ'.

- The deviation due to the fact that we don't approximate the first sub-ROOBP in the decomposition. Since the length of the first sub-ROOBP is βn, the deviation is bounded by $\beta \leq \frac{\delta}{2} = \delta'$.

Hence, our approximation deviates by a factor of at most $2\delta' = \delta$. We conclude with the following theorem, that summarizes what we achieved in this section.

Theorem A.25 *For $\beta \leq \frac{\delta}{2}$ and $\alpha = \Omega(\delta^2)$, δ-distance-approximation for (α, β, w)-sparse ROOBPs with error probability η reduces to $O(\frac{w^2}{\alpha})$ calls to $\frac{\delta}{2}$-distance-approximation for w-sparse ROOBPs with error probability $O(\frac{\alpha}{w^2} \cdot \eta)$.*

A.5 δ-Distance-Approximation Algorithm for an ROOBP of Constant Width

In this section, we present our main algorithm: a δ-distance-approximation algorithm for an ROOBP B of constant width (see Algorithm A.33). In our main algorithm, we use Algorithm A.22 to perform distance-approximation for not-too-many locally-sparse sub-ROOBPs of B. We now explain how we choose these sub-ROOBPs.

We first construct a decomposition for B, denoted the final decomposition $\mathcal{B}_f$, such that every $B_i \in \mathcal{B}_f$ is $((\frac{\delta}{2})^2, \frac{\delta}{2}, w)$-locally-sparse (see Definition A.19), and that δ-distance-approximation for B reduces to δ-distance-approximation for all $B_i \in \mathcal{B}_f$. We stress that in contrast to the decomposition we use in Section A.4, the dissection levels of $\mathcal{B}_f$ are not predefined, and so the sub-ROOBPs in $\mathcal{B}_f$ are not necessarily of the same size, and there might be many sub-ROOBPs in $\mathcal{B}_f$. Moreover, $\mathcal{B}_f$ is not necessarily connected. We show how to handle both issues in Sections A.5.3 and A.5.2, respectively.

We note that the local sparseness of the sub-ROOBPs in the decomposition is dependent only on the decomposition's dissection levels, and not on the specific sources and sinks we choose (this is because local sparseness only restricts pairs of w-connected levels, and the first and last levels of every sub-ROOBP (after dissection) contains only the source and the sink, respectively, and therefore are never part of a w-connected pair). Thus, we separate the construction of $\mathcal{B}_f$ to the following two steps:

- In Section A.5.1, we choose the dissection levels of $\mathcal{B}_f$. We allow ourselves to address $\mathcal{B}_f$ as a decomposition already at this step, even though $\mathcal{B}_f$ is not a valid decomposition according to Definition A.20 until we choose the source and sink for every sub-ROOBP in $\mathcal{B}_f$). This is because the selection of the specific source and sink does not affect the local sparsity of a sub-ROOBP. Then, we show that every $B_i \in \mathcal{B}_f$ is $((\frac{\delta}{2})^2, \frac{\delta}{2}, w)$-locally-sparse.
- In Section A.5.2, we choose the specific source and sink for each sub-ROOBP in $\mathcal{B}_f$ (according to the dissections levels we choose in the previous step) such that we don't lose too much in distance. That is, δ-distance-approximation for B reduces to δ-distance-approximation for all $B_i \in \mathcal{B}_f$.

Since the number of sub-ROOBPs in $\mathcal{B}_f$ might be large, we cannot afford to perform distance approximation for all the sub-ROOBPs in $\mathcal{B}_f$. To solve this, in Section A.5.3, we show how to choose a small set of sub-ROOBPs (taken from $\mathcal{B}_f$) such that the aforementioned reduction still holds.

Lastly, in Section A.5.4, we use the above to present our main algorithm: a δ-distance-approximation algorithm for an ROOBP of constant width.

A.5.1 Choosing the Dissection Levels of the Final Decomposition

In this section, we show how we choose the dissection levels $\mathcal{L} = (L_{r_1}, ..., L_{r_{h_f}})$ for $\mathcal{B}_f$, such that every $B_i \in \mathcal{B}_f$ is $((\frac{\delta}{2})^2, \frac{\delta}{2}, w)$-locally-sparse (see Definition A.19). Indeed, for every $B_i \in \mathcal{B}_f$, we need to show that the local sparsity conditions for B_i apply (with respect to the length of B_i).

Remark A.26 *Actually, we show that every $B_i \in \mathcal{B}_f$ that is **not too small** is $((\frac{\delta}{2})^2, \frac{\delta}{2}, w)$-locally-sparse. In the case that B_i **is** small, we can afford to perform distance approximation to B_i by simply reading the entire input for B_i (see more details in Section A.5.4).*

We choose the dissection levels $\mathcal{L}$ iteratively: we go through the levels of B, $(L_0, ..., L_n)$, in iterations, and in the l'th iteration, we decide whether L_l should be a dissection level for $\mathcal{B}_f$ based on some

greedy condition. We start with presenting the condition for choosing the first dissection level, L_{r_1}, and then we present the generalization for choosing any dissection level, L_{r_i}. We only prove correctness for the case of the first dissection level, since the proof of the other levels is very similar.

Definition A.27 (γ-good Level) *Let $\gamma > 0$. We say that L_l is γ-**good** if L_l is backwards-w-connected to some γl-close level.*

Now, assume L_l is the first γ-good level in B. Indeed, it must be that $l > 1$, because there is only one vertex in L_0 (the sink s), and therefore L_1 cannot be backwards-w-connected to L_0. Moreover, it must be that $l > \frac{1}{\gamma}$, since no level is w-connected to itself. In the next claim, we use L_l as the dissection level of B_1, and we assume that $l > \frac{1}{\gamma^2}$ (to make sure that any level $L_{l'}$ after $L_{\gamma l}$ has $l' > \frac{1}{\gamma}$). Then, we use the fact that every $L_{l'}$ in the range $\gamma l < l' < l$ is **not** γ-good, to show that B_1 is (γ^2, γ, w)-locally-sparse. Details follow.

Claim A.28 *Let L_l be the first level of B that is γ-good, and assume $l > \frac{1}{\gamma^2}$. If we choose the dissection level of B_1 as L_l, then B_1 is (γ^2, γ, w)-locally-sparse.*

Assuming Claim A.28, we can choose $\gamma = \frac{\delta}{2}$, and B_1 is $((\frac{\delta}{2})^2, \frac{\delta}{2}, w)$-locally-sparse.

Proof: First, we observe that Definition A.19 depends on the length of the ROOBP, and in our case the length of B_1 is l (since we choose L_l as the dissection level of B_1). That is, to show that B_1 is (γ^2, γ, w)-locally-sparse, we need to show that for every pair $L_{l'}, L_{l''}$ such that $\gamma l < l'' < l' < l$, if $L_{l'}$ and $L_{l''}$ are w-connected, then $L_{l'}$ and $L_{l''}$ are **not** $\gamma^2 l$-close. But this is true by the fact that we don't choose $L_{l'}$ as the dissection level of B_1. Then, we know that $L_{l'}$ is not γ-good, which by Definition A.27 implies that $L_{l'}$ is **not** backwards-w-connected to any $\gamma l'$-close level. Specifically, if $L_{l'}$ **is** backwards-w-connected to $L_{l''}$, then $l' - l'' \geq \gamma l' > \gamma^2 l$, and $L_{l'}$ and $L_{l''}$ are not $\gamma^2 l$-close. ■

We now generalize Claim A.28 to choose the dissection level for every $B_i \in \mathcal{B}_f$: We set $L_{r_0} = L_0$, and we assume we already chose dissection levels $(L_{r_1}, ..., L_{r_{i-1}})$. Then, if we choose L_l as the i'th dissection level for some $l > r_{i-1}$, then the length of B_i is $l - r_i$. This leads us to the following claim.

Definition A.29 ((γ, r_{i-1})-good Level) *Let $\gamma > 0$. We say that L_l is (γ, r_{i-1})-**good** if L_l is after $L_{r_{i-1}}$, and L_l is backwards-w-connected to some $\gamma(l - r_{i-1})$-close level.*

Claim A.30 *Let L_l be the first level of B that is (γ, r_{i-1})-good, and assume $l - l_{r_{i-1}} > \frac{1}{\gamma^2}$. If we choose the dissection level of B_i as L_l, then B_i is (γ^2, γ, w)-locally-sparse.*

Thus, to choose the dissection levels of $\mathcal{B}_f$, we do the following: We iterate through $l \in \{0, ..., n\}$, and for $i \geq 1$, we choose L_{r_i} as the first level which is $(\frac{\delta}{2}, r_{i-1})$-good. Then, by Claim A.30, every (not too small) $B_i \in \mathcal{B}_f$ is $((\frac{\delta}{2})^2, \frac{\delta}{2}, w)$-locally-sparse. Lastly, we add $L_n = L_{h_f}$ as the last dissection level of $\mathcal{B}_f$, to make sure $\mathcal{B}_f$ is a valid decomposition of B. Then, all levels of the last sub-ROOBP B_{h_f} are not $(\frac{\delta}{2}, h_f)$-good. Using a similar argument to the one we made in Claim A.28, this sub-ROOBP is either relatively small, or is $((\frac{\delta}{2})^2, \frac{\delta}{2}, w)$-locally-sparse.

A.5.2 Choosing the Sources and Sinks of the Final Decomposition

Let $\mathcal{L} = (L_{r_1}, ..., L_{r_{h_f}})$ be the dissection levels we choose for $\mathcal{B}_f$ according to the method described in Section A.5.1. That is, let $\mathcal{L}$ be the dissection levels we get by choosing the i'th dissection level

as the first level that is $(\frac{\delta}{2}, r_{i-1})$-good, and add $L_{r_{h_f}} = L_n$ to $\mathcal{L}$. Since we choose the dissection levels according to the structure of B, there could be many dissection levels in $\mathcal{L}$. Thus, we cannot afford to choose the source and sinks for all sub-ROOBPs in $\mathcal{B}_f$. To solve this, we show that we can choose the source and sink of each $B_i \in \mathcal{B}_f$ independently. That is, for $i \neq j$, the source and sink selection for B_i is not dependent on the source and sink selection for B_j.

Specifically, for each $B_i \in \mathcal{B}_f$, we choose the source and sink that yields the minimal distance of x from B_i. In Claim A.31, we show a lower bound for the distance of x from all the sub-ROOBPs in $\mathcal{B}_f$. That is, we show that, although the decomposition that we choose is not necessarily connected, we still don't lose too much in distance of x from B.

The high level idea is as follows: by the way we constructed $\mathcal{L}$, we know that for every $i \in [h_f - 1]$, the dissection level of B_i **is** backwards-w-connected to a relatively-close level (relatively to the length of B_i). This implies that the possible sinks of B_i are interchangeable: That is, we can change the sink of B_i to any other sink in that level without paying too much in distance. Specifically, we show that we can change the sink of B_i to the source of B_{i+1}, and get a new sub-ROOBP B'_i, such that the distance of B'_i from x is not "much" more than the distance of B_i from x. Then, if we do so for every B_i (but the last), we get that $(B'_1, ..., B'_{h_f-1}, B_{h_f})$ is a connected decomposition (see Figure 4). Since any connected decomposition is of distance at least ε from x, we know that our original decomposition, $\mathcal{B}_f$, cannot be too close to x. Details follow.

Claim A.31 *If for every $B_i \in \mathcal{B}_f$, we choose the source and sink that yields the minimal distance of x from B_i, then:*

$$\frac{\sum_i \Delta(x,B_i)}{n} \geq \varepsilon - \frac{\delta}{2}$$

Proof: For $i \in [h_f - 1]$, we make the following notations:

- Let $u_{r_{i-1}} \in L_{r_{i-1}}$ and $v_{r_i} \in L_{r_i}$ be the source and sink that yields the minimal distance of x from B_i.
- Let $x_i^{\min}$ be an accepting input for B_i which minimizes the distance to the input x
- Let $P_i^{\min}$ be the $u_{r_{i-1}} \rightsquigarrow v_{r_i}$ path associated with $x_i^{\min}$.
- Denote the length of B_i by $\rho_i = r_i - r_{i-1}$.

By the way we construct $\mathcal{L}$, we know that the dissection level of B_i **is** backwards-w-connected to some $\frac{\delta}{2}\rho_i$-close level, denoted $L_{r'_i}$. This means that any vertex in $L_{r'_i}$ is connected to any vertex in L_{r_i}. Specifically, denote the vertex in $P_i^{\min}$ that resides in $L_{r'_i}$ as $w_{r'_i}$, and we get that $w_{r'_i}$ is connected to u_{r_i} (the source of B_{i+1}). Therefore, there exists a $u_{r_{i-1}} \rightsquigarrow w_{r'_i} \rightsquigarrow u_{r_i}$ path, denoted $P_i'^{\min}$, that is different from $P_i^{\min}$ only in the last $r_i - r'_i \leq \frac{\delta}{2}\rho_i$ steps (at most). Denote the input associated with $P_i'^{\min}$ as $x_i'^{\min}$. Then, we can define a new sub-ROOBP $B'_i = B[u_{r_{i-1}}, u_{r_i}]$, for which $x_i'^{\min}$ is an accepting path. Thus, by the triangle inequality, we get:

$$\Delta(x, B_i) = \Delta(x, x_i^{\min}) \geq \Delta(x, x_i'^{\min}) - \Delta(x_i^{\min}, x_i'^{\min}) \geq \Delta(x, B'_i) - \frac{\delta}{2}\rho_i$$

If we do so for every $B_i \in \mathcal{B}_f$ (but the last), we get that $(B'_1, ..., B'_{h-1}, B_h) = (B'_1, ..., B'_h)$ is a connected decomposition, and so:

$$\sum_i \frac{\Delta(x,B_i)}{n} \geq \sum_i \frac{\Delta(x,B'_i) - \frac{\delta}{2}\rho_i}{n} \geq \sum_i \frac{\Delta(x,B'_i)}{n} - \frac{\delta}{2} \geq \varepsilon - \frac{\delta}{2}$$

■

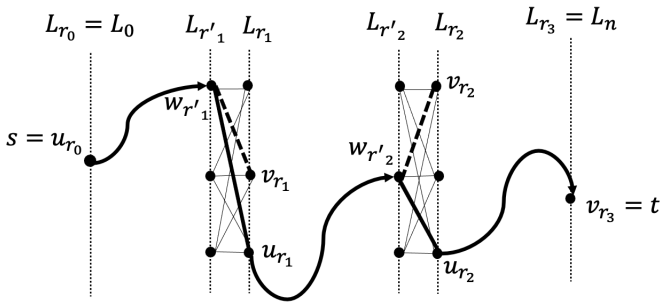

Figure 4: We can change the sink of every sub-ROOBP to the source of the next sub-ROOBP and get a connected decomposition, without paying too much in distance.

Remark A.32 *Indeed, for every B_i, we don't actually know which source and sink yield the minimal distance from x. However, we can approximate it by using Algorithm A.22 to perform distance approximation for every possible source-sink pair, and choose the one that yields the minimal distance approximation.*

A.5.3 Choosing sub-ROOBPs from the Final Decomposition

In the previous sections we show that every (not too small) $B_i \in \mathcal{B}_f$ is $((\frac{\delta}{2})^2, \frac{\delta}{2}, w)$-locally-sparse, so we can use Algorithm A.22 to perform distance approximation for it. In addition, we show that distance approximation for B reduces to distance approximation for every $B_i \in \mathcal{B}_f$. The problem is there could be many sub-ROOBPs in $\mathcal{B}_f$, so we cannot afford to perform distance approximation for **every** $B_i \in \mathcal{B}_f$.

Assuming our error probability parameter is η, we solve this by choosing $m = O(\frac{\log(\frac{1}{\eta})}{\delta^2})$ sub-ROOBPs from $\mathcal{B}_f$ with probability proportional to their length, and performing distance approximation for each of the chosen sub-ROOBPs (with every possible source and sink) using Algorithm A.22, with error probability parameter $\frac{\eta}{2mw^2}$. Then, for each sub-ROOBP, we take the minimal distance approximation obtained, and we output the average for all the chosen sub-ROOBPs (here, we take a simple average without length normalization, since the normalization is reflected in the length-proportional probability).

The above guarantees that with probability at least $1 - \frac{\eta}{2}$, **all** the calls to Algorithm A.22 are successful (i.e. for each sub-ROOBP, the distance approximation provided by Algorithm A.22 does not deviate by more than δ'). Then, assuming all the calls are successful, we can use Chernoff Bound to show that with probability at least $1 - \frac{\eta}{2}$, the average distance approximation for all the chosen sub-ROOBPs is a δ-distance approximation for B, and we get an overall success probability of $1 - \eta$.

A.5.4 The Actual Algorithm

Let B be an ROOBP of width w. In Sections A.5.1, A.5.2 and A.5.3, we show that if we do the below steps:

- Choose $\mathcal{B}_f$'s dissections levels as $(L_{r_1}, ..., L_{r_{h_f-1}}, L_n)$, such that for every $i \in [h_f - 1]$, L_{r_i} is the first level that is $(\frac{\delta}{2}, r_{i-1})$-good.

- Define the source and sink of every $B_i \in \mathcal{B}_f$ as the pair which yields the minimal distance of x from B_i.
- Independently choose $m = O(\frac{\log(\frac{1}{\eta})}{\delta^2})$ sub-ROOBPs from $\mathcal{B}_f$ with probability proportional to their length.

Then, if we output the average distance of all sub-ROOBPs chosen above, we get a δ-distance approximation for B. If $B_i \in \mathcal{B}_f$ is relatively small (i.e. $\rho_i \leq O(\frac{1}{\delta^2})$ where ρ_i is the length of B_i), then we can afford to read the entire input for B_i and compute the exact distance for the pair which yields the minimal distance from B_i, whereas if $\rho_i > O(\frac{1}{\delta^2})$, we can use Algorithm A.22 to approximate the distance for each source and sink, and choose the source and sink which yield the minimal distance approximation for B_i.

Indeed, we might "pay" in distance per the deviation that Algorithm A.22 may have. Therefore, we set a different deviation parameter (i.e. δ') for the recursive calls than our input parameter, δ. In addition, as explained in Section A.5.3, we need to set a different error probability parameter (i.e. η') for the recursive calls than our input error probability parameter, η. Details follow.

Algorithm A.33 *δ-Distance-Approximation Algorithm for an ROOBP of Constant Width*

- ***Input (Query Access)****: $x \in \{0,1\}^n$*
- ***Massive Parameter****: An ROOBP B of length n, width at most w, source vertex s and sink vertex t*
- ***Deviation Parameter****: $\delta > 0$*
- ***Error Probability****: $\eta > 0$*

1. *Define $\delta' = \frac{\delta}{3}$, $m = O(\frac{log(\frac{1}{\eta})}{\delta'^2}) = O(\frac{log(\frac{1}{\eta})}{\delta^2})$ and $\eta' = \frac{\eta}{2mw^2}$.*
2. *Iterate through the levels of B, and (sequentially) define the dissection levels $(L_{r_1}, ..., L_{r_{h_f-1}}, L_n)$ as follows: For every $i \in [h_f - 1]$, set L_{r_i} as the first level that is $(\frac{\delta'}{2}, r_{i-1})$-good. This can be done using Algorithm A.37 (see more details in Section A.7).*
3. *For m times, independently choose at random an index $i \in [h_f]$ with probability proportional to B_i's length (denoted $\rho_i = r_i - r_{i-1}$). Define the set of indices chosen as $I = (i_1, ..., i_m)$.*
4. *For every $i \in I$:*
   (a) *For every $u \in L_{r_{i-1}}, v \in L_{r_i}$, approximate the distance of x from $B[u,v]$ as follows:*
      i. *If $\rho_i < O(\frac{1}{\delta'^2})$, then determine the distance of x from $B[u,v]$ by reading the entire input for $B[u,v]$ (i.e. reading $x_i = x[r_{i-1}, r_i]$) and finding the minimal distance of x_i from an accepting input for $B[u,v]$ (i.e. finding the minimal $\Delta(x_i, x'_i)$ over all x'_i for which $B[u,v](x'_i) = 1$).*
      ii. *Otherwise (i.e. $\rho_i \geq O(\frac{1}{\delta'^2})$), apply Algorithm A.22 on the input x, with $B[u,v]$, deviation parameter δ' and error probability η'.*
   (b) *Define the minimum approximation obtained in the previous step as $\hat{\varepsilon}_i$.*
5. ***Return*** *$\frac{1}{m}\sum_i \hat{\varepsilon}_i$*

Remark A.34 *The conditions of Step 2 depend only on the structure of B and therefore can be tested without any query access to the input x. Thus, the query complexity of Algorithm A.33 depends only on the query complexity of Step 4(a). The query complexity of Step 4(a)(ii) depends only on the query complexity of Algorithm A.22 and on the number of recursive calls we make to Algorithm A.22, which is bounded by $m \cdot w^2 = O(\frac{w^2 log(\frac{1}{\eta})}{\delta^2})$. The query complexity of Step 4(a)(i) is upper bounded by $m \cdot w^2 \cdot O(\frac{1}{\delta^2}) = O(\frac{w^2 \cdot log(\frac{1}{\eta})}{\delta^4})$. Since $w > 1$, and we make at least $O(\frac{log(\frac{1}{\eta})}{\delta^2})$ queries in the base algorithm, the number of queries of Step 4(a)(ii) dominates the number of queries of Step 4(a)(i).*

We show the algorithm above is a δ-approximation algorithm for an ROOBP of width w. Indeed, as shown in Section A.5.1, every sub-ROOBP in Step 4(b) is $((\frac{\delta'}{2})^2, \frac{\delta'}{2}, w)$-locally-sparse, therefore we can apply Algorithm A.22 on it. Then, our estimation only deviates from ε by the following additive factors:

1. The deviation due to the fact that we use only m sub-ROOBPs. Then, since we choose I independently, and we choose each $i \in I$ with probability proportional to the sub-ROOBP's length, we can use Chernoff's bound to show that with probability at least $1-\frac{\eta}{2}$, this deviation is at most δ'.

2. The deviation due to the fact that we use Algorithm A.22 to find the minimal distance approximation for every sub-ROOBP. As shown in Section A.5.3, since we use Algorithm A.22 with deviation parameter δ' and error probability parameter η', we know that with probability at least $1 - \frac{\eta}{2}$ our approximation for every sub-ROOBP deviates from the minimal-distance one by at most δ'.

3. The deviation due to the fact that we use the minimal approximation in each sub-ROOBP **independently**, out of every possible source-sink pair (this could cause us to under-estimate the distance, since we don't enforce the decomposition to be connected). By Claim A.31, since we choose the parameter for the dissection levels as $\gamma = \frac{\delta'}{2}$, this deviation is at most $\frac{\delta'}{2}$.

Hence, with probability at least $1-\eta$, our approximation deviates by a factor of at most $\delta' + \delta' + \frac{\delta'}{2} < \delta$. We conclude with the following theorem, that summarizes what we achieved in this section.

Theorem A.35 *δ-distance-approximation for ROOBPs of width w with error probability η reduces to $O(\frac{w^2 log(\frac{1}{\eta})}{\delta^2})$ calls to $\frac{\delta}{3}$-distance-approximation for $((\frac{\delta}{6})^2, \frac{\delta}{6}, w)$-locally-sparse ROOBPs with error probability $O(\frac{\delta^2}{w^2 \cdot log(\frac{1}{\eta})} \cdot \eta)$.*

A.6 Computational Complexity (of Algorithm A.33)

We start with calculating the query complexity of Algorithm A.33, which is our main complexity measure. We show that it is independent of the length of the input x. Additionally, if we assume the width of the ROOBP and the error probability are constant, we show that the query complexity of Algorithm A.33 is polynomial in the distance parameter, δ.

Then, we finish with calculating the time complexity of Algorithm A.33, which we show is linear in the input length, n.

A.6.1 Query Complexity

We start by summarzing what we have showed so far:

- In Theorem A.35, we showed that δ-distance-approximation for ROOBPs of width w with error probability η reduces to $O(\frac{w^2\log(\frac{1}{\eta})}{\delta^2})$ calls to $\frac{\delta}{3}$-distance-approximation for $((\frac{\delta}{6})^2, \frac{\delta}{6}, w)$-locally-sparse ROOBPs with error probability $O(\frac{\delta^2}{w^2\cdot\log(\frac{1}{\eta})}\cdot\eta)$.
- In Theorem A.25, we showed that δ-distance-approximation for (α, β, w)-sparse ROOBPs (with $\alpha = O(\delta^2)$ and $\beta \leq \frac{\delta}{2}$) with error probability η reduces to $O(\frac{w^2}{\alpha})$ calls to $\frac{\delta}{2}$-distance-approximation for w-sparse ROOBPs with error probability $O(\frac{\alpha}{w^2}\cdot\eta)$.
- In Theorem A.17, we showed that δ-distance-approximation for w-sparse ROOBPs with error probability η reduces to $O(\frac{w^2}{\delta})$ calls to $\frac{\delta}{2}$-distance-approximation for ROOBPs of width $w-1$ with error probability $O(\frac{\delta}{2w^2}\cdot\eta)$.

Hence, δ-distance-approximation for ROOBP of width w with error probability η reduces to $\text{poly}(\frac{\log(\frac{1}{\eta})\cdot w}{\delta})$ calls to $\frac{\delta}{12}$-distance-approximation for ROOBP of width $w-1$ with error probability $\text{poly}(\frac{\delta}{w\cdot\log(\frac{1}{\eta})})\cdot\eta$.

Letting $\mathrm{Q}_w(\delta,\eta)$ denote the query complexity of Algorithm A.33, we get that:

$$\mathrm{Q}_w(\delta,\eta) = \text{poly}(\tfrac{\log(\frac{1}{\eta})\cdot w}{\delta})\cdot\mathrm{Q}_{w-1}(\tfrac{\delta}{12}, \text{poly}(\tfrac{\delta}{w\cdot\log(\frac{1}{\eta})})\cdot\eta)$$

Then, for every η' used throughout the algorithm, we have $\eta' = O((\frac{\delta}{\log(\frac{1}{\eta})\cdot 2^w})^{O(w)}\cdot\eta)$, thus we get:

$$\text{poly}(\tfrac{\log(\frac{1}{\eta'})\cdot w}{\delta}) = \text{poly}(\tfrac{\log(\frac{1}{\eta})\cdot w}{\delta})$$

Thus, using $\mathrm{Q}_1(\delta,\eta) = O(\frac{\log(\frac{1}{\eta})}{\delta^2})$, the solution to the recursion yields:

$$\mathrm{Q}_w(\delta) = \textstyle\prod_{i\in[w]}\text{poly}(\frac{\log(\frac{1}{\eta})\cdot w}{\delta/12^{i-1}}) = (\frac{\log(\frac{1}{\eta})\cdot 2^w}{\delta})^{O(w)}$$

Indeed, if we assume η and w are constant, we get a query complexity that is polynomial in $\frac{1}{\delta}$.

A.6.2 Time Complexity

As shown in Section A.6.1, we call each of our algorithms for at most $(\frac{\log(\frac{1}{\eta})\cdot 2^w}{\delta})^{O(w)}$ times. Let us review the computations that our algorithms make when initiated:

- In Algorithm A.15, we look at at most $\frac{w^2}{\delta}$ pairs of distance $\delta\cdot n$, and for each pair, we check whether the pair is connected and narrowing. Checking the connectivity of all pairs can be done in time $O(w^2\cdot n)$, whereas checking whether all pairs are narrowing is done by using Algorithm A.37, which gives time complexity of $O(2^w\cdot w\cdot n)$.
- In Algorithm A.22, we look at at most $O(\frac{w^2}{\delta^2})$ pairs, of distance $\delta^2\cdot n$ and for each pair, we check whether the pair is connected. Again, checking the connectivity of all pairs can be done in time $O(w^2\cdot n)$.

- In Algorithm A.33, we use Algorithm A.37 to define the dissection levels of the final decomposition, which gives time complexity of $O(2^w \cdot w \cdot n)$.

All the above yields time complexity of $O(2^w \cdot w \cdot n)$. We need to performs these computations every time we call one of the algorithms above, since B might be different in every such call. Using Algorithm A.37 (see Remark A.38), the overall time complexity of Algorithm A.33 is upper-bounded by $(\frac{\log(\frac{1}{\eta}) \cdot 2^w}{\delta})^{O(w)} \cdot n$.

A.7 Finding Significant Levels

Given a ROOBP B of width w, the task we face is to find, for every level L_j in B, a maximum $i < j$ such that L_i has w vertices and each vertex in L_i is connected to each vertex in L_j (or determine that no such i exists).

The key observation is that it suffices to determine for each vertex $v \in L_j$ a maximum $i < j$ such that L_i has w vertices and each vertex in L_i is connected to v (or determine that no such i exists). Whenever such a level exists, we call it the significant level of v and denote it by $\sigma(v)$; that is, for $v \in L_j$, we denote by $\sigma(v)$ the maximal $i < j$ such that L_i has w vertices and each vertex in L_i is connected to v. If no such level exists, we define $\sigma(v) = 0$. Our observation that it suffices to determine the significant levels of all vertices is supported by the following claim.

Claim A.36 (significant levels and connectivity) *Let $i \stackrel{\text{def}}{=} \min_{v \in L_j}\{\sigma(v)\} > 0$. Then, every vertex in L_j is connected to all vertices in L_i.*

Hence, the foregoing task is solved by finding, for each $j \in [n]$, the value of $\min_{v \in L_j}\{\sigma(v)\}$.

Proof: Let i be as in the hypothesis, $u \in L_j$ be such that $\sigma(u) = i$, and consider an arbitrary $v \in L_j$ such that $\sigma(v) > i$. Then, the paths connecting u to all vertices in L_i pass through $L_{\sigma(v)}$, and so their prefixes can be used to connect v to all vertices in L_i (i.e., for every $x \in L_i$, the path from u to x passes through some $w \in L_{\sigma(v)}$, and yields a path from v to w and then to x). ■

We extend the notion of significant levels to subsets of vertices as follows. For a non-empty set $S \subseteq L_j$, the significant level of S, denoted $\sigma(S)$, is the maximum $i < j$ such that L_i has w vertices and each vertex in L_i is connected to some vertex in S (and $\sigma(S) = 0$ if no such i exists). Letting $t \in [n]$ be the minimum level that has w vertices, we find the significant levels of all subsets of all levels $j > t$.

Algorithm A.37 (finding significant levels) *We proceed in iterations from $j = t+1$ to $j = n$. For $j \in [t+1, n]$ and every non-empty $S \subseteq L_j$, we proceed as follows.*

1. *Determine the set S' of vertices in L_{j-1} that are connected to some vertices in S; that is, S' is the maximal set such that for every $v' \in S'$ there exists a vertex $v \in S$ that has an incoming edge from v'.*
2. *If $|S'| = w$, then let $\sigma(S) \leftarrow j - 1$.*
3. *Otherwise* (i.e., $|S'| < w$), *let $\sigma(S) \leftarrow \sigma(S')$.*

It can be shown by induction on j that for every non-empty $S \subseteq L_j$ the setting of $\sigma(S)$ is indeed correct. Evidently, the algorithm has running time $O(n \cdot 2^w \cdot w)$, whereas the results we seek are the significant levels of all singletons.

Remark A.38 *We use the algorithm above in Algorithms A.15 and A.33, as follows:*

- *For Algorithm A.15, we can determine whether a pair (u, v) is narrowing by checking if $\sigma(u) = 0$ and $\sigma(v) > 0$. This is immediate from Claim A.13.*
- *For Algorithm A.33, we can determine whether a level L_{r_i} is (γ, r_{i-1})-good (see Def A.29) by checking that L_{r_i} has w vertices, and that $\min_{v \in L_{r_i}}\{\sigma(v)\} > r_i - \gamma(r_i - r_{i-1})$. This is immediate from the key observation at the beginning of the current section.*