A Structural Theorem for Local Algorithms with Applications to Coding, Testing, and Verification*

Marcel Dall'Agnol
University of Warwick
msagnol@pm.me

Tom Gur†
University of Cambridge
tom.gur@cl.cam.ac.uk

Oded Lachish
Birkbeck, University of London
o.lachish@bbk.ac.uk

Abstract

We prove a general structural theorem for a wide family of local algorithms, which includes property testers, local decoders, and PCPs of proximity. Namely, we show that the structure of every algorithm that makes q adaptive queries and satisfies a natural robustness condition admits a sample-based algorithm with $n^{1-1/O(q^2 \log^2 q)}$ sample complexity, following the definition of Goldreich and Ron (TOCT 2016). We prove that this transformation is nearly optimal. Our theorem also admits a scheme for constructing privacy-preserving local algorithms.

Using the unified view that our structural theorem provides, we obtain results regarding various types of local algorithms, including the following.

- We strengthen the state-of-the-art lower bound for relaxed locally decodable codes, obtaining an *exponential* improvement on the dependency in query complexity; this resolves an open problem raised by Gur and Lachish (SICOMP 2021).
- We show that any (constant-query) testable property admits a sample-based tester with sublinear sample complexity; this resolves a problem left open in a work of Fischer, Lachish, and Vasudev (FOCS 2015), bypassing an exponential blowup caused by previous techniques in the case of adaptive testers.
- We prove that the known separation between proofs of proximity and testers is essentially maximal; this resolves a problem left open by Gur and Rothblum (ECCC 2013, Computational Complexity 2018) regarding sublinear-time delegation of computation.

Our techniques strongly rely on relaxed sunflower lemmas and the Hajnal–Szemerédi theorem.

Keywords: local algorithms, sample-based algorithms, coding theory, property testing, adaptivity, sunflower lemmas.

*An extended abstract of this paper appeared in SODA 2021 as "A Structural Theorem for Local Algorithms with Applications to Coding, Testing, and Privacy" [DGL21].

†Tom Gur is supported by the UKRI Future Leaders Fellowship MR/S031545/1 and EPSRC New Horizons Grant EP/X018180/1.

Contents

1 Introduction

Sublinear-time algorithms are central to the theory of algorithms and computational complexity. Moreover, with the surge of massive datasets in the last decade, understanding the power of computation in sublinear time rapidly becomes crucial for real-world applications. Indeed, in recent years this notion received a great deal of attention, and algorithms for a plethora of problems were studied extensively.

Since algorithms that run in sublinear time cannot even afford to read the entirety of their input, they are forced to make decisions based on a small local view of the input and are thus often referred to as *local algorithms*.[1] Prominent notions of local algorithms include *property testers* [RS96, GGR98], which are probabilistic algorithms that solve approximate decision problems by only probing a minuscule portion of their input; *locally decodable codes* (LDCs) [KT00] and *locally testable codes* (LTCs) [GS06], which are codes that admit algorithms that, using a small number of queries to their input, decode individual symbols and test the validity of the encoding, respectively; and *probabilistically checkable proofs* (PCPs) [FGL+91, AS98, ALM+98], which are encodings of NP-proofs that can be verified by examining only (say) 3 bits of the proof.

While the foregoing notions are often grouped under the umbrella term of local algorithms, they are in fact very distinct. Indeed, local algorithms perform fundamentally different tasks, such as testing, self-correcting, decoding, computing a local function, or verifying the correctness of a proof. Moreover, the tasks are often performed under different promises (e.g., proximity to a valid codeword in the case of LDCs, and deciding whether an object has a property or is far from having it in the case of property testing).

Nevertheless, despite the aforementioned diversity, one of our main *conceptual* contributions is capturing a fundamental structural property that is common to all of algorithms above and beyond, which in turn implies sufficient structure for obtaining our main result. We build on work of Fischer, Lachish and Vasudev [FLV15] as well as Gur and Lachish [GL21], which imply an essentially equivalent structure for *non-adaptive* testers and local decoders, respectively; our generalisation captures both and extends beyond them to the adaptive setting, as well as to other classes of algorithms.

More specifically, we first formalise the notion of local algorithms in the natural way: we define them simply as probabilistic algorithms that compute some function $f(x)$, with high probability, by making a small number of queries to the input x. We then observe that, except for degenerate cases, having a promise on the input is necessary for algorithms that make a sublinear number of queries to it. Finally, we formalise a natural robustness condition that captures this phenomenon and is shared by most reasonable interpretations of local algorithms.

1.1 Robust local algorithms

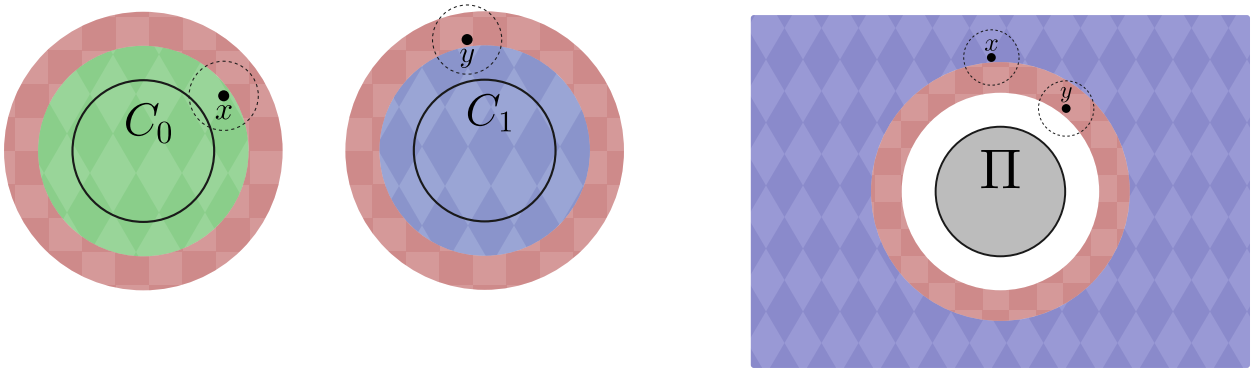
We say that a local algorithm is *robust* if its output is stable under minor perturbations of the input.[2] To make the discussion more precise, we define a (ρ_0, ρ_1)-*robust local algorithm* M for computing a partial function $f\colon \mathcal{P} \to \{0,1\}$ (where $\mathcal{P} \subset \{0,1\}^n$) as a local algorithm that satisfies the following: for every input w that is ρ_0-close to x (which may or may not belong to $\mathcal{P}$) such

[1] This terminology makes explicit that the (sublinear) parameter of interest is the algorithm's query complexity, as opposed to, say, space (as in the streaming model) or time.

[2] Note that the notion of robustness is a priori orthogonal to locality; however, as robustness is arguably the main structural property of local algorithms, we restrict the discussion to their intersection.

that $f(x) = 0$, we have $M^w = 0$ with high probability; and, for every w that is ρ_1-close to x such that $f(x) = 1$, we have $M^w = 1$ w.h.p. We remark that our results extend to larger alphabets (see Section 4, where we formally define robust local algorithms).

We illustrate the expressivity of robust local algorithms via two examples: property testing and locally decodable codes. We remark that similarly, locally testable codes, locally correctable codes, relaxed LDCs, PCPs of proximity, and other notions can all be cast as robust local algorithms (see Sections 4.2 to 4.4).



(a) A local decoder for the code C with decoding radius δ. Codewords whose i^{th} message bit equals 0 (resp. 1) for a fixed i comprise C_0 (resp. C_1). The decoder is robust in the $\delta/2$-neighbourhood of C, with tiled blue and green patterns. Inputs in the checkerboard red area are within distance δ from C, but their $\delta/2$-neighbourhoods are not.

(b) An ε-tester for property Π. Inputs in the tiled blue area are 2ε-far from Π, where the tester is robust. While inputs in the checkerboard red area are rejected by the tester, this is not necessarily the case for their ε-neighbourhoods.

Figure 1: Casting local decoders and property testers as robust local algorithms.

Locally decodable codes. An LDC is a code that admits algorithms for decoding each individual bit of the message of a moderately corrupted codeword; that is, a code $C\colon \{0,1\}^k \to \{0,1\}^n$ with *decoding radius* δ for which there exists a probabilistic algorithm D that, given an index $i \in [k]$, queries a string w promised to be δ-close to a codeword $C(x)$ and outputs $D^w(i) = x_i$ with high probability.

Observe that D can be viewed as a $(\delta/2, \delta/2)$-robust local algorithm for local decoding with respect to decoding radius $\delta/2$, that is, for the function $f\colon [k] \times \mathcal{P} \to \{0,1\}$ where $\mathcal{P} = B_{\delta/2}(\operatorname{Im} C)$ is the $\delta/2$-neighbourhood of C.[3] This is because the $\delta/2$-neighbourhood of any point that is $\delta/2$-close to a codeword is still within the decoding radius, and thus the algorithm decodes the same value when given either a codeword or a string in its neighbourhood; see Fig. 1a.

Property testing. Property testers are algorithms that solve approximate decision problems by only probing a minuscule part of their input, and are one of the most widely studied types of

[3]Note that f additionally receives a coordinate $i \in [k]$ as explicit input, which is allowed by the formal definition of robustness (see Definition 4.2).

sublinear algorithms (see, e.g., the textbook [Gol17]).

An ε-tester T for a property Π queries a string x and, with high probability, outputs 1 if $x \in \Pi$, and outputs 0 if x is ε-far from Π. Here, unlike with local decoders, there is no robustness at all with respect to 1-inputs:[4] we can only cast T as an $(\varepsilon, 0)$-robust local algorithm for the function $f\colon \mathcal{P} \to \{0,1\}$ where $\mathcal{P}$ is the union of Π and the complement of its 2ε-neighbourhood. We refer to such robustness as *one-sided*.

Note that while T does not ε-test Π robustly, it is robust when viewed as a 2ε-tester: doubling the proximity parameter ensures the ε-neighbourhood of each 0-input is still rejected (see Fig. 1b).

By the previous discussion, our scope includes local algorithms that only exhibit *one-sided* robustness. Accordingly, we define *robust local algorithms* (without specifying the robustness parameters) as (ρ_0, ρ_1)-robust local algorithms where $\max\{\rho_0, \rho_1\} = \Omega(1)$.[5] We stress that while dealing with one-sided robustness is significantly more technically involved, our results also hold for this type of local algorithm.

1.2 Main result

In this work, we capture structural properties that are common to all robust local algorithms and leverage it to obtain a transformation that converts them into (uniform) *sample-based algorithms*.

Sample-based algorithms are provided with uniformly distributed labeled samples, or alternatively, query each coordinate independently with some fixed probability. Adopting the latter perspective, the *sample complexity* of such an algorithm is the expected number of coordinates that it samples.

In the following, we use n to denote the input size and assume the alphabet Σ over which the input is defined is not too large (e.g., $|\Sigma| \leq n^{1/q^4}$ suffices).

Theorem 1 (Theorem 6.1, informally stated)**.** *Every robust local algorithm with query complexity q can be transformed into a* sample-based *local algorithm with sample complexity $n^{1-1/O(q^2 \log^2 q)}$.*

We stress that the robustness in Theorem 1 is only required on *part of the input space* (i.e., need only be one-sided); indeed, otherwise the structural properties captured become much more restrictive (and are not shared by, e.g., property testers).

Moreover, we prove that the transformation in Theorem 1 is optimal up to a quadratic factor in the dependency on the query complexity; that is, q-query robust local algorithms cannot be transformed into sample-based algorithms with sample complexity $n^{1-1/o(q)}$ (see Section 7.3 for a more precise statement).

Our proof of Theorem 1 strongly relies on analysing the query behaviour of robust local algorithms by partitioning their local views into relaxed sunflowers and using volume lemmas that are implied by their robustness. We build on the Hajnal–Szemerédi theorem to analyse sampling from relaxed sunflowers (see Section 2 for a detailed technical overview).

By the generality of our definition, we can apply Theorem 1 to a wide family of well-studied algorithms such as locally testable codes, locally decodable and correctable codes, relaxed LDCs, universal LTCs, PCPs of proximity, and more (see Section 4 for details on how to cast these algorithms as robust local algorithms).

[4] Unless the tester is *tolerant*: an $(\varepsilon_1, \varepsilon_2)$-tolerant tester is, by definition, ε_1-robust with respect to its 1-inputs.

[5] Although Theorem 1 assumes an algorithm satisfying this condition, we remark that a weaker one suffices. Supposing (without loss of generality) that $\rho_0 \geq \rho_1$, only *a single input* x must imply $M^w = 0$ when w is $\Omega(1)$-close to x; then the result follows even for $\rho_0 = \Theta(n^{-1/q}) = o(1)$, where q is the query complexity of M.

We note that [FLV15] and [GL21] obtain an essentially equivalent transformation for testers and decoders, respectively, through "lossy" versions of our relaxed sunflower lemmas (they extract *one* relaxed sunflower from the local views, rather than partition them) that applies to non-adaptive algorithms; by a trivial transformation from adaptive to non-adaptive algorithms that incurs an exponential increase in the query complexity, these previous works show transformations whose sample-based algorithms have complexity $n^{1-1/\exp(q)}$, which we reduce to $n^{1-1/\mathrm{poly}(q)}$ (indeed, as far down as $n^{1-1/\tilde{O}(q^2)}$).

Motivation. The notion of sample-based local algorithms was first defined in [GGR98], and its systematic study was initiated by Goldreich and Ron [GR16]. This is an intrinsically interesting model of computation with practical potential, as obtaining random samples is much easier than implementing full query access to a large input. Moreover, sample-based local algorithms admit schemes for multi-computation (i.e., simultaneously computing multiple functions of the input using the same queries), as well as schemes for private local computation, on which we elaborate below.

Private local computation. We wish to highlight an interesting application of Theorem 1 to privacy. Suppose a client wishes to compute a function of data that is stored on a server, e.g., decode a symbol of a code or test whether the data has a certain property. Typically, the query behaviour of a local algorithm may leak information on which function the client attempts to compute. However, since sample-based algorithms probe their input uniformly, they can be used to compute the desired function without revealing any information on which function was computed, e.g., which coordinate was decoded or which property was tested.

Furthermore, since Theorem 1 transforms any robust local algorithm into a sample-based local algorithm that probes its input *obliviously to the function it computes*,[6] then (after standard error-reduction) we can apply Theorem 1 to *many algorithms at once* and reuse the samples to obtain a local algorithm that computes multiple functions at the same time (see Section 1.3.2).

1.3 Applications

We proceed to our main applications, which range over three fields of study: coding theory, property testing and probabilistic proof systems. We remark that our testing application follows as a direct corollary of Theorem 1, but adapting it to decoders and proof systems require additional arguments.

1.3.1 Relaxed locally decodable codes

Locally decodable codes play an important role in contemporary coding theory. Since their systematic study was initiated by Katz and Trevisan [KT00], they made a profound impact on several areas of theoretical computer science (see, e.g., [Tre04, Yek12, KS17] and references therein), and led to practical applications in distributed storage [HSX+12].

Despite the success and attention that LDCs received in the last two decades, the best construction of $O(1)$-query LDCs has *super-polynomial* blocklength (cf. [Efr12], building on [Yek08]). This barrier led to the study of *relaxed* LDCs, which were introduced in the foundational work of Ben-Sasson, Goldreich, Harsha, Sudan, and Vadhan [BGH+06]. In a recent line of research, relaxed LDCs

[6] We stress that such obliviousness does not *not* correspond to a differential privacy guarantee.

and variants thereof were applied to PCPs [MR10, DH13, RR20], property testing [CG18], data structures [CGdW13] and probabilistic proof systems (e.g., [GR17, DGRMT22]).

Loosely speaking, this relaxation allows the local decoder to abort on a small fraction of the indices, yet crucially, still avoid errors. More accurately, a *relaxed* LDC $C\colon \{0,1\}^k \to \{0,1\}^n$ with *decoding radius* δ is a code that admits a probabilistic algorithm, a decoder, which on input index $i \in [k]$ makes queries to a string $w \in \{0,1\}^n$ that is δ-close to a codeword $C(x)$ and satisfies the following: (1) if the input is a valid codeword (i.e., $w = C(x)$), the decoder outputs x_i with high probability; and (2) otherwise, with high probability, the decoder must either output x_i or a special "abort" symbol $\perp$, indicating it detected an error and is unable to decode.[7]

This seemingly modest relaxation allows for obtaining dramatically stronger parameters. Indeed, [BGH+06] constructed a q-query relaxed LDC with blocklength $n = k^{1+1/\Omega(\sqrt{q})}$, and raised the problem of whether it is possible to obtain better rates; the best known construction, obtained in recent work of Asadi and Shinkar [AS21], improves it to $n = k^{1+1/\Omega(q)}$. We stress that *proving lower bounds on relaxed LDCs is significantly harder than on standard LDCs*, and indeed, the first non-trivial lower bound was only recently obtained in [GL21], which shows that, to obtain query complexity q, the code must have blocklength

$$n \geq k^{1+\frac{1}{O(2^{2q}\cdot \log^2 q)}}.$$

This shows that $O(1)$-query relaxed LDCs cannot obtain quasilinear length, a question raised in [Gol11], but still leaves exponential room for improvement in the dependency on query complexity (note that even for $q = O(1)$ this strongly affects the asymptotic behaviour). Indeed, eliminating this exponential dependency was raised as the main open problem in [GL21].

Fortunately, our technical framework is general enough to capture *relaxed* LDCs as well, and in turn, our main application for coding theory resolves the aforementioned open problem by obtaining a lower bound with an *exponentially* better dependency on the query complexity. Along the way, we also extend the lower bound to hold for relaxed decoders with *two-sided error*, resolving another problem left open in [GL21].

Theorem 2 (Corollary 7.8, informally stated)**.** *Any relaxed LDC $C\colon \{0,1\}^k \to \{0,1\}^n$ with constant decoding radius δ and query complexity q must have blocklength at least*

$$n \geq k^{1+\frac{1}{O(q^2 \log^2 q)}}.$$

This also makes significant progress towards resolving the problem due to [BGH+06], by narrowing the gap between lower and upper bounds to merely a quadratic factor.

1.3.2 Property testing

Recall that a standard ε-tester for a property Π is endowed with the ability to make queries, accepting inputs in Π and rejecting inputs that are ε-far from Π. An alternative definition, first given in [GGR98], only provides the tester with uniformly distributed labeled samples (or, equivalently, with uniform and independent queries to each coordinate). Such algorithms are called *sample-based testers* and have received attention recently [GR16, FGL14, BGS15, FLV15, CFSS17, BMR19a, BMR19b].[8]

[7] As observed in [BGH+06], these two conditions suffice for obtaining a third condition which guarantees that the decoder only outputs $\perp$ on an arbitrarily small fraction of the coordinates.

[8] More accurately, these are *uniform* sample-based testers (in contrast to the [BGS15] tester, which queries coordinates in a random subspace). We adopt the original terminology of [GR16] for simplicity.

As an immediate corollary of Theorem 1, we obtain that any constant-query testable property (up to $\sqrt[5]{\log n}$-query, in fact) admits a sample-based tester with sublinear sample complexity. This resolves a problem left open in a work of Fischer, Lachish, and Vasudev [FLV15], who showed a similar statement for non-adaptive testers, by extending it to the adaptive setting.[9]

Theorem 3 (Corollary 7.1, informally stated)**.** *Any property $\Pi \subseteq \{0,1\}^n$ that is ε-testable with q queries admits a sample-based 2ε-tester with sample complexity $n^{1-1/O(q^2 \log^2 q)}$.*

This also admits an application to *adaptive multi-testing*, where the goal is to simultaneously test a large number of properties. In Section 4.2 we show that as a corollary of Theorem 3 we can multi-test, with sublinear query complexity, exponentially many properties, namely $k = \exp\left(n^{1/\omega(q^2 \log^2 q)}\right)$, that are each testable with q adaptive queries.

1.3.3 Proofs of proximity

Proofs of proximity [RVW13] are probabilistic proof systems that allow for delegation of computation in sublinear time. They were studied extensively in recent years, finding applications in cryptography with both computational [KR15] and information-theoretic security [RRR21, BRV18].

In the non-interactive setting, we have a verifier that wishes to ascertain the validity of a given statement, using a short (sublinearly long) explicitly given proof, and a sublinear number of queries to its input. Since the verifier cannot even read the entire input, it is only required to reject inputs that are far from being valid. Thus, the verifier is only assured of the proximity of the statement to a correct one. Such proof systems can be viewed as the NP (or, more accurately, MA) analogue of property testing, and are referred to as MA proofs of proximity (MAPs).

As such, one of the most fundamental questions regarding proofs of proximity is their relative strength in comparison to testers; that is, whether verifying a proof for an approximate decision problem can be done significantly more efficiently than solving it. One of the main results in [GR18] is that this can indeed be the case. Namely, there exists a property Π which: (1) admits an adaptive MAP with proof length $O(\log n)$ and query complexity $q = O(1)$; and (2) requires at least $n^{1-1/\Omega(q)}$ queries to be tested without access to a proof.[10]

In Section 7.3 we use Theorem 1 to show that the foregoing separation is nearly tight.

Theorem 4 (Theorem 7.11, informally stated)**.** *Any property $\Pi \subseteq \{0,1\}^n$ that admits an adaptive MAP with query complexity q and proof length p also admits a tester with query complexity $p \cdot n^{1-1/O(q^2 \log^2 q)}$.*

Interestingly, we remark that we rely on Theorem 4 to prove the (near) optimality of Theorem 1 (see Section 7.3 for details).

1.4 Discussion and open questions

Our work leaves several interesting directions and open problems that we wish to highlight. Firstly, we stress that our structural theorem is extremely general, and indeed the robustness condition that induces the structure required by Theorem 1 appears to hold for most reasonable interpretations

[9] [FLV15] applies to adaptive testers with an exponential blowup in query complexity, which our result avoids.

[10] We remark that the bound in [GR18] is stated in a slightly weaker form. However, it is straightforward to see that the proof achieves the bound stated above. See Section 7.3.

of robust local algorithms. While we gave applications to coding theory, property testing, and probabilistic proof systems, it would be interesting to see whether our framework (or a further generalisation of it) could imply applications to other families of local algorithms, such as PAC learners, local computation algorithms (LCAs), and beyond.

Open question 1. Can Theorem 1 and the framework of robust local algorithms be used to obtain query-to-sample transformations for PAC learners and LCAs?

One promising direction that we did not explore is on rate lower bounds on PCPs of proximity (PCPPs). Such bounds are notoriously hard to get, and indeed the only such bounds we are aware of are those in [BHLM09], which are restricted to special setting of 3-query PCPPs. We remark that our framework captures PCPPs, and that in light of the rate lower bounds it allowed us to obtain for relaxed LDCs, it seems feasible to obtain rate lower bounds on PCPPs as well.

Open question 2. Can we obtain rate lower bounds on q-query PCPs of proximity for $q > 3$?

Another interesting question involves the optimality of our transformation. Recall that Theorem 1 transforms q-query robust local algorithms into sample-based local algorithms with sample complexity $n^{1-1/O(q^2 \log^2 q)}$, whereas in Section 7.3 we show that any such transformation must yield an algorithm with sample complexity $n^{1-1/\Omega(q)}$. This still leaves a quadratic gap in the dependency on query complexity. We remark that closing this gap could lead to fully resolving an open question raised in [BGH+06] regarding the power of relaxed LDCs.

Open question 3. What is the optimal sample complexity obtained by a transformation from robust local algorithms to sample-based local algorithms?

Note, moreover, that we focus on query (or sample) complexities and provide a computationally inefficient transformation, iterating over exponentially many input strings; the *computational* cost of such transformations is an interesting problem in its own regard.

Open question 4. Are there efficient transformations from robust to sample-based algorithms?

Finally, in Section 1.2 we discuss an application of our main result to privacy-preserving local computation, where one can compute one out of a large collection of functions without revealing information regarding which function was computed.[11] While Theorem 1 implies such a scheme that only requires probing the input in sublinear locations, the number of probes is quite high. Moreover, by the near-tightness of our result, we cannot expect a significant improvement of this scheme.

Nevertheless, we find it very interesting to explore whether for structured families of functions (as, for example, admitted by the canonical tester for dense graphs in [GT03]), or for statistical or computational notions of privacy (i.e., where the distributions of queries obtained from each function are statistically close, or indistinguishable to a computationally bounded adversary), the query complexity of this scheme can be significantly reduced.

Open question 5 (Private testing of small families)**.** Do there exist schemes for private local computation with small query complexity?

[11] Borrowing terminology from zero-knowledge, we can call this notion of privacy *perfect* – as opposed to statistical or computational – as the query pattern is completely independent from the choice of function.

Organisation

The rest of the paper is organised as follows. In Section 2, we provide a high-level technical overview of the proof of our main result and its applications. In Section 3, we briefly discuss the preliminaries for the technical sections. In Section 4, we present our definition of robust local algorithms and show how to cast various types of algorithms in this framework. In Section 5, we provide an arsenal of technical tools, including relaxed sunflower lemmas and a sampling lemma that builds on the Hajnal–Szemerédi theorem. In Section 6, we use the foregoing tools to prove Theorem 1. Finally, in Section 7, we derive our applications to coding theory, property testing, and proofs of proximity.

2 Technical overview

In this section, we outline the techniques used and developed in the course of proving Theorem 1 and its applications. Our techniques build on and simplify ideas from [FLV15, GL21], but are significantly more general and technically involved, and in particular, offer novel insight regarding adaptivity in local algorithms.

Our starting point, which we outline in Section 2.1, generalises the techniques of [GL21] (which are, in turn, inspired by [FLV15]) to the setting of robust local algorithms. Then, in Section 2.2, we identify a key technical bottleneck in previous works: *adaptivity*. We discuss the fundamental challenges that adaptivity imposes, and in Section 2.3 we present our strategy for meeting these challenges and the tools that we develop for dealing with them, as well as describe our construction. Subsequently, in Section 2.4, we provide an outline of the analysis of our construction, which relies on the Hajnal–Szemerédi theorem to sample from structured set systems we call *daisies*.

The setting. Recall that our goal is to transform a robust (query-based) local algorithm into a sample-based algorithm with sublinear sample complexity. Towards this end, let M be a (ρ_0, ρ_1)-robust local algorithm for computing a function $f\colon \{0,1\}^n \to \{0,1\}$.[12] Since we also need to deal with *one-sided robustness*, assume without loss of generality that $\rho_1 = 0$ and $\rho := \rho_0 = \Omega(1)$. Recall that the algorithm M receives query access to a string $x \in \{0,1\}^n$, flips at most r random coins, makes at most q queries to this string and outputs $f(x) \in \{0,1\}$ with probability at least $1 - \sigma$.

For simplicity of exposition, we assume that the error rate is $\sigma = \Theta(1/q)$, the query complexity is constant ($q = O(1)$), and the randomness complexity r is bounded by $\log(n) + O(1)$. We remark that the analysis trivially extends to non-constant values of q, and that we can achieve the other assumptions via simple transformations, which we provide in Section 5.4, at the cost of logarithmic factors in q. In the following, our aim is to construct a *sample-based* local algorithm N for computing the function f, with sample complexity $O\big(n^{1-1/2q^2}\big) = n^{1-1/O(q^2)}$.

2.1 The relaxed sunflowers method

As a warm-up, we first suppose that the algorithm M is *non-adaptive*. (This section gives an overview of the techniques of [GL21], which suffice in the non-adaptive case.) Then we can simply represent M as a distribution μ over a collection of query sets $\mathcal{S}$, where each $S \in \mathcal{S}$ is a subset of

[12] In general, the function f may depend on an explicitly given parameter (e.g., an index for decoding in the case of relaxed LDCs), but for simplicity of notation, we omit this parameter in the technical overview.

$[n]$ of size q, and predicates $\{f_S \colon \{0,1\}^q \to \{0,1\}\}_{S \in \mathcal{S}}$, as follows. The algorithm M draws a set $S \in \mathcal{S}$ according to μ, queries S, obtains the *local view* $x_{|S}$ (i.e., x restricted to the coordinates in S), and outputs $f_S(x_{|S})$.

Consider an algorithm N that samples each coordinate of the string x independently with probability $p = 1/n^{1/2q^2}$ (and aborts in the rare event that this exceeds the desired sample complexity).[13] Naively, we would have liked N^x to emulate an invocation of the algorithm M by sampling the restriction of x to a query set $S \sim \mu$.

Indeed, if the distribution μ is "well spread", the probability of obtaining such a local view of M is high. Suppose, for instance, that all of the query sets are pairwise disjoint. In this case, the probability of N sampling any particular local view is p^q, and we expect N to obtain $\Omega(p^q n) = \Omega(n^{1-1/2q})$ local views (recall that the support size of μ, i.e., the number of query sets, is $O(n)$ by our assumption of $\log n + O(1)$ randomness complexity). However, if μ is concentrated on a small number of coordinates, it is highly unlikely that N will obtain a local view of M. For example, if M queries the first coordinate of x with probability 1, then we can obtain a local view of M with probability at most p, which is negligible.

Fortunately, we can capitalise on the robustness condition to deal with this problem. We first illustrate how to do so for an easy special case, and then deal with the general setting.

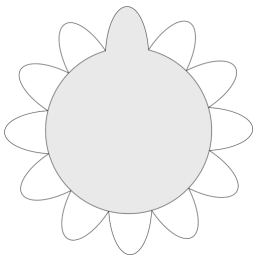
Special case: sunflower query set. Suppose that μ is concentrated on a small coordinate set K and is otherwise disjoint, i.e., the support of μ is a *sunflower* with kernel K of size at most ρn; see Fig. 2a. Since the query sets are disjoint outside of K, by the discussion above we will sample many sets except for the coordinates in K (i.e., sample the petals of the sunflower). Recall that if x is such that $f(x) = 0$, then the $(\rho, 0)$-robust algorithm M outputs 0, with high probability, on any input y that is ρ-close to x. Thus, even if we arbitrarily assign values to K and use them to complete sampled petals into full local views, we can emulate an invocation of M that will output as it would on x.

If *all inputs* in the promise of M were robust (as is the case for LDCs, but *not* for testers, relaxed LDCs,[14] and PCPPs), then the above would suffice. However, recall that we are not ensured robustness when x is such that $f(x) = 1$. To deal with that, we can enumerate over all possible assignments to the kernel K, considering the local views obtained by completing sampled petals into full local views by using each kernel assignment to fill in the values that were not sampled. Observe that: (1) when the input x is a 1-input and N considers the kernel assignment that coincides with x, a majority of local views (a fraction of at least $1 - \sigma$) will lead M^x to output 1; and (2), when x is a 0-input, a minority of local views (a fraction of at most σ) will lead M^x to output 1 *under any kernel assignment*.

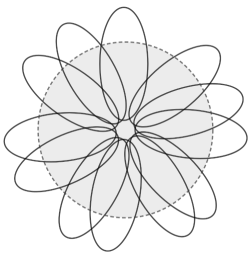
The sample-based algorithm N thus outputs 1 if and only if it sees, for some kernel assignment, a majority of local views that lead M to output 1. Recall that there is asymmetry in the robustness of M (while 0-inputs are robust, 1-inputs are not), which translates into asymmetric output conditions for N. Note, also, that correctness of this procedure for 0-inputs requires that not even a single kernel assignment would lead N to output incorrectly; but our assumption on the error rate ensures that the probability of sampling a majority of petals whose local views will lead to an error is sufficiently small to tolerate a union bound over all kernel assignments, as long as $|K|$ is small

[13] This choice of p will be made clear in Section 2.4; see Footnote 21.

[14] The type of robustness that relaxed LDCs admit is slightly more subtle, since it deals with a larger alphabet that allows for outputting $\perp$. See discussion in Section 7.2.



(a) A *sunflower* with one of its sets shaded. The intersection of any two sets results in the same set, the kernel.



(b) A *daisy* with its kernel shaded, whose boundary is the dashed line. Outside the kernel each point is covered by a bounded number of petals.

Figure 2: Sunflowers and daisies.

enough.

General case: extracting a heavy daisy from the query sets. Of course, the combinatorial structure of the query sets of a local algorithm is not necessarily a sunflower and may involve many complex intersections. While we could use the *sunflower lemma* to extract a sunflower from the collection of query sets, we stress that the size of such a sunflower is *sublinear*, which is not enough in our setting (as we deal with constant error rate).

Nevertheless, we can exploit the robustness of M even if its query sets only have the structure of a relaxed sunflower, referred to as a *daisy*, with a small kernel. Loosely speaking, a t-daisy is a sunflower in which the kernel is not necessarily the intersection of all petals, but is rather a small subset such that every element outside the kernel is contained in at most t petals;[15] see Fig. 2b (and see Section 5.1 for a precise definition).

Using a *daisy lemma* [FLV15, GL21], we can extract from the query sets (the support of μ) of the robust local algorithm M a t-daisy $\mathcal{D}$ with t roughly equal to $n^{i/q}$ and a kernel K of size roughly $n^{1-i/q}$, where $i \in [q]$ bounds the size of the petals of $\mathcal{D}$. Moreover, the *weight* $\mu(\mathcal{D}) = \sum_{S \in \mathcal{D}} \mu(S)$ is significantly larger than the error rate σ of M (recall that we assumed a sufficiently small $\sigma = \Theta(1/q)$). Thus, even if the daisy contains all local views that lead to an error, their total weight would still be small with respect to that of local views leading to a correct decision; hence, the query sets in the daisy $\mathcal{D}$ well-approximate the behaviour of M, and we can disregard the sets in the support of μ that do not belong to $\mathcal{D}$ at the cost of a negligible increase to the error rate.

Crucially, the intersection bound t implies that *sampling a daisy is similar to sampling a sunflower*: since petals do not intersect heavily, with high probability many of them are fully queried (as is the case with sunflowers). The bound on $|K|$, on the other hand, allows us to implement the sampling-based algorithm we discussed for the sunflower case, except with respect to a daisy. The kernel is sufficiently small so that the output of M is unchanged under any assignment to K, and suffices to tolerate a union bound when considering all possible assignments to K.

It follows that the daisy $\mathcal{D}$ provides enough "sunflower-like" structure for the sample-based algorithm N defined previously to succeed, with high probability, when it only considers the query sets in $\mathcal{D}$ and enumerates over all assignments to its kernel.

[15] In Definition 5.1, a t-daisy has t as a function from $[q]$ to $\mathbb{N}$ and allows for a tighter bound on the number of intersecting petals. We use the simplified definition of [GL21] in this technical overview.

2.2 The challenge of adaptivity

Let us now attempt to apply the transformation laid out in the previous section to a robust local algorithm M that makes q *adaptive* queries. In this case, M may choose to query distinct coordinates depending on the answers to its previous queries, and thus *there is no single distribution* μ that captures its query behaviour.

Observe that now, rather than inducing a distribution on sets, the algorithm M induces a distribution over *decision trees* of depth q, as the behaviour of a randomised query-based algorithm M^x can be described by choosing a decision tree according to its random string, then performing the adaptive queries according to the evaluation of that tree on the input $x \in \{0,1\}^n$. By our assumption on the randomness complexity of M, this distribution is supported on $\Theta(n)$ decision trees. Note that for any *fixed* input x, the decision tree collapses to a path, and hence the distribution over decision trees induces a distribution over query sets, which we denote μ_x (see Fig. 3).

A naive way of transitioning from decision trees to sets is by querying all of the branches of each decision tree. Alas, doing so would increase the query complexity of M exponentially from q to (more than) 2^q, which would in turn lead to a sample-based algorithm with a much larger sample complexity than necessary. Thus, we need to deal with the far more involved structure induced by distributions over decision trees, which imposes significant technical challenges. For starters, since our technical framework inherently relies on a combinatorial characterisation of algorithms, we first need to find a method of transitioning from decision trees to (multi-)sets without increasing the query complexity of the local algorithm M.

To this end, a key idea is to enumerate over all random strings and their corresponding decision trees, and extract all q-sets (i.e., sets of size q) corresponding to each branch of each tree. This leaves us with a combinatorial *multi-set* $\mathcal{S}$ (as multiple random strings may lead to the same decision tree, and branches of distinct decision trees may query the same set) with $\Theta(2^q \cdot n) = \Theta(n)$ query sets, of size q each, corresponding to all possible query sets induced by all possible input strings.[16] Note that $\mathcal{S}$ contains the elements of the support of μ_x for all inputs $x \in \{0,1\}^n$ and that, for any fixed input x, the vast majority of these query sets may not be relevant to this input: each $S \in \mathcal{S} \setminus \mathrm{supp}(\mu_x)$ corresponds to a branch of a decision tree that the bits of x would have not led to query.

This already poses a significant challenge to our approach, as we would have liked to extract a heavy daisy $\mathcal{D}$ from the collection $\mathcal{S}$ which well-approximates the query sets of M *independently of any input*. However, it could be the case that the sets that are relevant to an input x (i.e., $\mathrm{supp}(\mu_x)$) induce a completely different daisy (with potentially different kernels over which we'll need to enumerate) than the relevant sets for a different input y that differs from x on the values in the kernel, and so it is not clear at all that there exists a single daisy that well-approximates the query behaviour of the adaptive algorithm M for all inputs.

Furthermore, the above also causes problems with the kernel enumeration process. For each assignment κ to the kernel K, denote by $x_\kappa \in \{0,1\}^n$ the word that takes the values of κ in K and the values of x outside of K. Recall that the crux of our approach is to simulate executions of M^{x_κ}, for each kernel assignment κ, using the values of the sampled petals and plugging in the kernel assignment to complete these petals into local views (assignments to full query sets). Hence, since relevant sets corresponding to different kernel assignments may be distinct, it is unclear how to rule according to the local views that each of them induce.

[16] We remark that this treatment of multi-sets allows us to significantly simplify the preparation for combinatorial analysis that was used in previous works involving sunflowers and daisies.

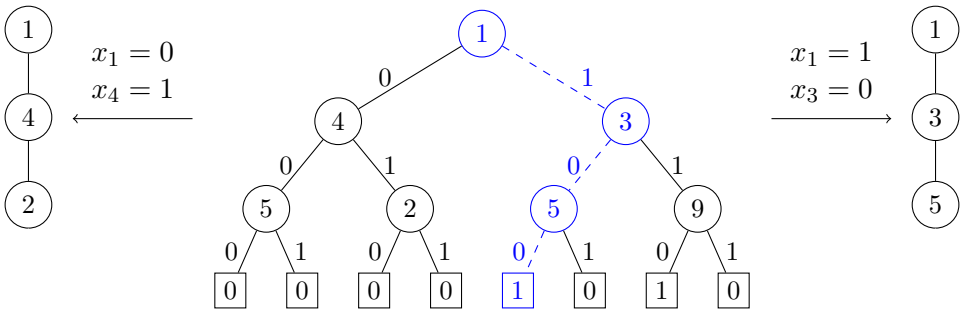


Figure 3: Decision tree of a 3-local algorithm. When the input x is such that $x_1 = 1$, $x_3 = 0$ and $x_5 = 0$, the branch highlighted in blue (and dashed) queries $\{1, 3, 5\}$ and outputs 1. When $x_1 = 0$ and $x_4 = 1$, this tree *induces* the query set $\{1, 2, 4\}$; when $x_1 = 1$ and $x_3 = 0$, it induces the set $\{1, 3, 5\}$. This "collapsing" of the query behaviour is illustrated on either side of the tree.

We overcome these challenges in the next section with a more sophisticated extraction of daisies that, crucially, *does not discard any query sets* of the adaptive algorithm M. Specifically, we will partition the (multi)-collection of all possible query sets into a collection of daisies and *simultaneously analyse all daisies in the partition* to capture the adaptive behaviour of the algorithm.

2.3 Capturing adaptivity in daisy partitions

Relying on techniques from [FLV15, GL21], we can not only extract a single heavy daisy, but rather *partition* a (multi-)collection of query sets into a family of daisies, with strong structural properties on which we can capitalise. This allows us to apply our combinatorial machinery without dependency on a particular input, and *analyse all daisies simultaneously.*

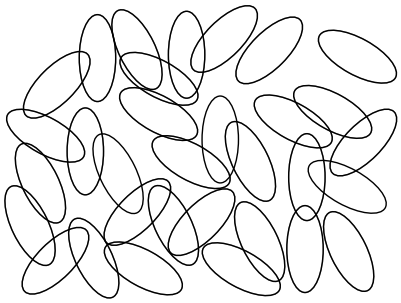
Daisy partition lemma. A refinement of the daisy lemma in [GL21], which we call a *daisy partition lemma* (Lemma 5.2), partitions a multi-set $\mathcal{S}$ of q-sets into $q + 1$ daisies $\{\mathcal{D}_i : 0 \leq i \leq q\}$ (see Fig. 4) with the following structural properties.

1. $\mathcal{D}_1$ is a $n^{1/q}$-daisy, and for $i > 1$, each $\mathcal{D}_i$ is a t-daisy with $t = n^{(i-1)/q}$;
2. The kernel K_0 of $\mathcal{D}_0$ coincides with that of $\mathcal{D}_1$, and, for $i > 0$, the kernel K_i of $\mathcal{D}_i$ satisfies $|K_i| \leq q|\mathcal{S}| \cdot n^{-i/q}$;
3. The petal $S \setminus K_i$ of every $S \in \mathcal{D}_i$ has size exactly i.

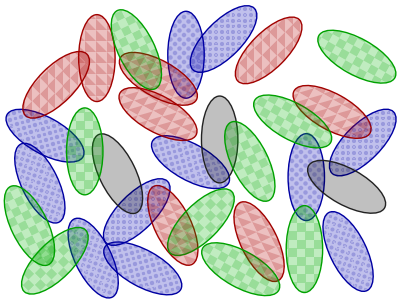
Moreover, the kernels form an incidence chain $K_q = \varnothing \subseteq K_{q-1} \subseteq \cdots \subseteq K_1 = K_0$. Note that $\mathcal{D}_0$ is vacuously a t-daisy for any t, since its petals are empty; and that our assumption on the randomness complexity of M implies $|K_i| = O(n^{1-i/q})$ when $i > 0$.

We may thus apply the daisy partition lemma to $\mathcal{S}$ and assert that, *for any input* x, there exists *some* $i \in \{0, \ldots, q\}$ such that $\mu_x(\mathcal{D}_i)$ is larger than $1/q$ (recall that, for all x, the support of μ_x is contained in $\mathcal{S}$); that is, each input may lead to a different heavy daisy, but there will always be at least one daisy that well-approximates the behaviour of the algorithm on input x. Alas, with only a local view of the input word, we are not able to tell which daisies are heavy and which are not.

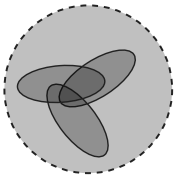
It is clear, then, that a sample-based algorithm that makes use of the daisy partition has to rule not only according to a single daisy, but rather according to all of them. But *how* exactly it



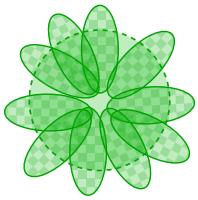
(a) A collection $\mathcal{S}$ of 3-sets before being partitioned.



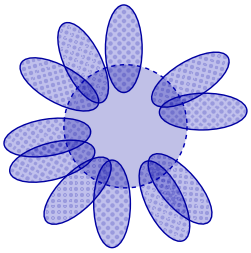
(b) The collection $\mathcal{S}$ partitioned into 4 daisies: $\mathcal{D}_0$ (shaded flat, grey), $\mathcal{D}_1$ (checkerboard, green), $\mathcal{D}_2$ (dotted, blue) and $\mathcal{D}_3$ (triangle pattern, red).



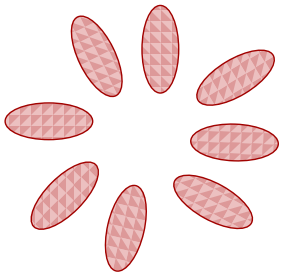
(c) $\mathcal{D}_0$, whose sets are entirely contained in the kernel K_0.



(d) $\mathcal{D}_1$ with $K_1 = K_0$, where each $S \in \mathcal{D}_1$ has a petal $S \setminus K_1$ of size 1.



(e) $\mathcal{D}_2$ with $K_2 \subseteq K_1$, where each $S \in \mathcal{D}_2$ has a petal $S \setminus K_2$ of size 2.



(f) $\mathcal{D}_3$, with (empty) kernel $K_3 = \varnothing$. Each query set $S \in \mathcal{D}_3$ has a petal $S \setminus K_3 = S$ of size 3.

Figure 4: Daisy partition.

should do so is a nontrivial question to answer, given that there are multiple daisies (and kernels) potentially interfering with one another.

Adaptivity in daisy partitions. A natural approach for dealing with multiple daisies simultaneously is by enumerating over every assignment to *all* kernels (i.e., to $\cup_i K_i$) and, for each such assignment, obtaining local views from all daisies and ruling according to the aggregated local views. Note that the incidence chain structure implies that enumerating over assignments to K_0 suffices, since each assignment to K_0 induces assignments to K_i for all i.

However, this approach leads to fundamental difficulties. Recall that correctness of the sample-based algorithm on 0-inputs depends on *no kernel assignment* causing an output of 1. Although for any assignment to K_i this happens with sufficiently small probability to ensure it is unlikely to happen on all $2^{|K_i|}$ assignments simultaneously, this does not hold true for assignments to larger kernels. More precisely, since $|K_{i-1}|$ may be larger than $|K_i|$ by a factor of $n^{1/q}$, an error rate that is preserved by $2^{|K_i|}$ assignments becomes unbounded if the number of assignments increases to $2^{|K_{i-1}|}$. This leads us to only consider, for query sets in $\mathcal{D}_i$, assignments to K_i rather than to the union of all kernels.

Put differently, we construct an algorithm that deals with each daisy independently, and whose correctness follows from a delicate analysis that aggregates local views taken from all daisies, which we outline in Section 2.4. We begin by considering a sample-based local algorithm N that extends

the strategy we used for a single daisy as follows. On input $x \in \{0,1\}^n$, it:

(1) samples each coordinate of the string x independently with probability $p = 1/n^{1/2q^2}$;
(2) for each $i \in \{0, \ldots, q\}$ and each assignment κ to the kernel K_i of the daisy $\mathcal{D}_i$, outputs 1 if a majority of local views leads M to output 1; and
(3) outputs 0 if no such majority is ever found.

First, note that since the algorithm N is constructed in a *white-box* manner, it has access to the description of all decision trees induced by the query-based algorithm M. Hence N^x is able to determine which local views correspond to a valid execution of M. Denoting by Q the set of coordinates that were sampled, an assignment κ to K_i induces, for each query set $S \subset Q \cup K_i$, the assignment $x_{\kappa|S}$; the sample-based algorithm N can check whether each such S is a relevant query set (i.e., belongs to the support of μ_{x_κ}) by verifying it arises from some branch of a decision tree of M that x_κ would have led to query. This allows N to ignore the non-relevant query sets and overcome the difficulty pointed out in the previous section.[17]

However, we remain with the issue that motivated searching for heavy daisies in the first place: there is no guarantee that every $\mathcal{D}_i$ well-approximates the algorithm M on all inputs. This is due to the use of relative estimates: if x is a 0-input and $\mu_x(\mathcal{D}_i)$ is smaller than the error rate σ, even when N considers the correct kernel assignment $x_{|K_i}$ with respect to x, it may find a majority of local views that leads M^x to output 1; indeed, nothing prevents all the "bad" query sets, which lead M^x to erroneously output 1, from being placed in the same daisy $\mathcal{D}_i$.

The solution is to use a simpler decision rule: absolute rather than relative. We count the number of local views leading to an output of 1, outputting 1 if and only if it crosses a threshold. The upper bound σ on the weight of "bad" query sets limits their number, and a large enough threshold prevents them from causing an incorrect output even if *no local view* leads to the correct one. Note that a different threshold τ_i is needed for each daisy $\mathcal{D}_i$, since the probability of sampling petals decreases as i increases. The thresholds τ_i must thus be carefully set to take this into account.

Finally, note that whenever the daisy $\mathcal{D}_0$ leads to an output of 1, this happens (almost) *independently of the input*: the assignment to every $S \in \mathcal{D}_0$ is determined solely by the assignment to K_0, because $S \subset K_0$. Therefore, the sample-based algorithm N disregards $\mathcal{D}_0$ in its execution.

The algorithm. By the discussion above, we obtain the following description for the sample-based algorithm N^x (with some parameters that we will set later).

1. Sample each coordinate of x independently with probability $p = 1/n^{1/2q^2}$. If the number of samples exceeds the desired sample complexity, abort.

2. For every $i \in [q]$ and every assignment κ to K_i, perform the following steps.

   (a) Count the number of sets in $\mathcal{D}_i$ with local views that lead M to output 1, which are *relevant for the assignment* κ and the queried values. If $i = 1$, discard the sets whose petals are shared by at least α local views.[18]

[17] We remark that in the accurate description of our construction (see Section 6.1), we capture all the information contained in the decision trees via *tuples* that contain, besides the query set, the assignment that led to it being queried as well as the output of the algorithm when it does so. The daisy partition lemma then allows to partition these tuples based on the structure of the sets they contain.

[18] The extra condition for $i = 1$ is necessary to deal with the looser intersection bound $t = n^{1/q} > n^{(i-1)/q}$ on $\mathcal{D}_1$. We discuss this in the next section.

(b) If the number is larger than the threshold τ_i, output 1.

3. If every assignment to every kernel failed to trigger an output of 1, then output 0.

In the next section we will present key technical tools that we develop and apply to analyse this algorithm, as well as discuss the parameters $\tau_i = \gamma_i \cdot np^i$ (where $\gamma_i = \Theta(1)$) and $\alpha = \Theta(1)$, and show it indeed suffices for the problem we set out to solve.

2.4 Analysis using a volume lemma and the Hajnal–Szemerédi theorem

To establish the correctness of the aforementioned sample-based algorithm, we shall first need two technical lemmas about sampling daisies. We will then proceed to provide an outline of the analysis of our algorithm.

2.4.1 Two technical lemmas

We sketch the proofs of two simple, yet important technical lemmas that will be paramount to our analysis: (1) a lemma that allows us to transition from arguing about probability mass to arguing about combinatorial volume; and (2) a lemma that allows us to efficiently analyse sampling petals of daisies with complex intersection patterns.

The volume lemma. We start by showing how to derive from the probability mass of query sets (i.e., the probability under μ_x when the input is x) a bound on the volume that the union of these query sets cover. This is provided by the following *volume lemma*, which captures what is arguably *the defining structural property of robust local algorithms*.

Recall that the sample-based algorithm N uses the query sets of a $(\rho, 0)$-robust algorithm M with error rate σ, which comprise the support of the distributions μ_x for all inputs x. Intuitively, these sets cannot be too concentrated (i.e., cover little volume), as otherwise slightly corrupting a word (in less than ρn coordinates) could require M to output differently, a behaviour that is prevented by the robustness of M. This intuition is captured by the following *volume lemma*.

Lemma 2.1 (Lemma 5.6, informally stated)**.** *Let $x \in \{0,1\}^n$ be a non-robust input (a 1-input in our case) and $\mathcal{S}$ be a subcollection of query sets in the support of μ_x. If $\mathcal{S}$ covers little volume (i.e., $\cup\mathcal{S} < \rho n$), then it has small weight (i.e., $\mu_x(\mathcal{S}) < 2\sigma$).*

We stress that the *robustness of the 0-inputs yields the volume lemma for 1-inputs.*[19] Note that the contrapositive of the volume lemma yields a desirable property for our sample-based algorithm: for any (non-robust) 1-input x, the query sets in $\mathrm{supp}(\mu_x)$ must cover a large amount of volume, so that we can expect to sample many such sets.

The Hajnal–Szemerédi theorem. Once we establish that a daisy covers a large volume, it remains to argue how this affects the probability of sampling petals from this large daisy, which is a key component of our algorithm. Recall that sampling the petals of a sunflower is trivial to do. However, with the complex intersection patterns that the petals of a daisy could have, we need a tool to argue about sampling petals of daisies.

[19] This is a rather subtle consequence of adaptivity; in the nonadaptive setting a symmetric volume lemma for b-inputs can be shown using robustness on b-inputs, for $b \in \{0,1\}$.

First, recall that the daisy partition lemma ensures that each $\mathcal{D}_i$ is a t-daisy where $t = n^{\max\{1,i-1\}/q}$, for all i. Observe that if $\mathcal{D}_i$ is a 1-daisy (which we call a *simple* daisy), that is, each point outside the kernel K_i is contained in at most one set $S \in \mathcal{D}_i$, then the sets in $\mathcal{D}_i$ have pairwise disjoint petals, so sampling them is *exactly* like sampling petals of a sunflower: these petals are sampled independently from one another, and we expect their number to be concentrated around the expectation of $p^i|\mathcal{D}_i|$ (recall that all petals have size i).

Of course, there is no guarantee that $\mathcal{D}_i$ is a simple daisy, though we expect it to *contain* a simple daisy if it is large enough. Indeed, greedily removing intersecting sets yields a simple daisy of size $\Theta(|\mathcal{D}_i|/t)$, but this does not suffice for our purposes because most of the sets in $\mathcal{D}_i$ are discarded.

Instead, we rely on the *Hajnal–Szemerédi theorem* to obtain a "lossless" transition from a t-daisy to a collection of simple daisies, from which sampling petals is easy. The Hajnal–Szemerédi theorem shows that for every graph G with m vertices and maximum degree Δ, and for any $k \geq \Delta + 1$, there exists a k-colouring of the vertices of G such that every colour class has size either $\lfloor m/k \rfloor$ or $\lceil m/k \rceil$. By applying this theorem to the incidence graph of the *petals* of query sets (i.e., the graph with vertex set $\mathcal{D}_i$ where we place an edge between S and S' when $(S \cap S') \setminus K_i \neq \varnothing$), which satisfies $\Delta(G) \leq 2t$ (see Claim 5.3) we obtain a partition of $\mathcal{D}_i$ into t simple daisies of the same size (up to an additive error of 1), and hence obtain stronger sampling bounds.

2.4.2 Analysis

Note that the probability that N samples too many coordinates (thus aborts) is exponentially small, hence we assume hereafter that this event did not occur.

We proceed to sketch the high-level argument of the correctness of the sampled-based algorithm N, described in the previous section, making use of tools above. This follows from two claims that hold with high probability: (1) *correctness on non-robust inputs*, which ensures that when x is a 1-input (i.e., is non-robust), there exists $i \in [q]$ such that when N considers the kernel assignment $x_{|K_i}$ (which coincides with the input), the number of local views that lead to output 1 crosses the threshold τ_i; and (2) *correctness on robust inputs*, which, on the other hand, ensures that when x is a 0-input (i.e., is robust), for *every kernel* K_i and *every kernel assignment*, the number of local views that lead to output 1 *does not cross* the threshold τ_i.

In the following, we remind that when the sample-based algorithm N considers a particular assignment κ to a kernel and counts the number of local views that lead to output 1, the algorithm only considers views that are *relevant* to x_κ (the input x where the values of its kernel are replaced by κ); that is, local views that arise from some branch of a decision tree of the adaptive algorithm M that would have led it to query these local views. While N does not know all of x, after collecting samples from x and considering the kernel assignment κ, it can check which local views are relevant to x_κ (see discussion in Section 2.3).

Correctness on non-robust inputs. We start with the easier case, where x is a non-robust input (in our case, $f(x) = 1$). We show that there exists $i \in [q]$ such that when N considers the kernel assignment $x_{|K_i}$, the number of local views that lead to output 1 crosses the threshold $\tau_i = \gamma_i \cdot np^i$. We begin by recalling that N disregards the daisy $\mathcal{D}_0$, whose query sets are entirely contained in the kernel K_0, and arguing that this leaves sufficiently many query sets that lead to output 1. Indeed, while we could not afford this if $\mathcal{D}_0$ was heavily queried by M given the 1-input x (i.e., if $\mu_x(\mathcal{D}_0)$ is close to $1 - \sigma$), an application of the volume lemma shows this is not the case: since $|K_0| = o(n)$, this volume is smaller than ρn, implying $\mu_x(\mathcal{D}_0) < 2\sigma$ *for all* 1*-inputs* x.

Apart from $\mathcal{D}_0$, the query sets in the daisy $\mathcal{D}_1$ whose petals are shared by at least α local views (for a parameter α to be discussed shortly) are also discarded, and we need to show that the loss incurred by doing so is negligible as well. This is accomplished with a slightly more involved application of the volume lemma: since the sets of $\mathcal{D}_1$ have petals of size 1, the subcollection $\mathcal{C} \subseteq \mathcal{D}_1$ of sets that are discarded covers a volume of at most $|K_1| + |\mathcal{C}|/\alpha$. For a sufficiently large choice of a constant $\alpha > 0$, we have $|\mathcal{C}|/\alpha \leq \rho n/2$ (recall that $\mathcal{C} \subseteq \mathrm{supp}(\mu_x)$ and $|\mathrm{supp}(\mu_x)| = \Theta(n)$ by the assumption on the randomness complexity of M). Since $|K_1| = o(n)$ and in particular $|K_1| < \rho n/2$, applying the volume lemma to $\mathcal{C}$ shows that $\mu_x(\mathcal{C}) < 2\sigma$.

Finally, the total weight of all query sets in $\mathrm{supp}(\mu_x)$ that lead to output 0 is at most σ (by definition of the error rate σ). This implies that the subcollection of $\mathrm{supp}(\mu_x)$ that leads to output 1 *and is not disregarded* has weight at least $1 - 2\sigma - 2\sigma - \sigma = 1 - 5\sigma$, and, for a sufficiently small value of σ (recall that $\sigma = \Theta(1/q)$), we have $1 - 5\sigma \geq 1/2$.

We now shift perspectives, and in effect use the volume lemma in the contrapositive direction: large weights imply large volumes. By a simple averaging argument, it follows that at least one daisy $\mathcal{D}_i$ has weight at least $(1 - 5\sigma)/q \geq 2\sigma$, and thus, by the volume lemma, *covers at least ρn coordinates*. Therefore, since $|\mathrm{supp}(\mu_x)| = \Theta(n)$ and μ_x *is uniform over a multi-collection of query sets*, this daisy contains $\Theta(n)$ "good" sets (that lead to output 1 and were not discarded). For the analysis, using the Hajnal–Szemerédi theorem, we partition the t-daisy $\mathcal{D}_i$ into t *simple* daisies of size $\Theta(n/t)$. Each such simple daisy has *disjoint* petals of size i, so that $\Omega(np^i/t)$ petals will be sampled except with probability $\exp\big(-\Omega(np^i/t)\big)$. Finally, this implies that, by setting $\gamma_i = \Theta(1)$ small enough, *when N considers the kernel assignment $x_{|K_i}$ to K_i*, at least $\tau_i = \gamma_i \cdot np^i$ petals are sampled except with probability

$$O(t) \cdot \exp\left(-\Omega\left(\frac{np^i}{t}\right)\right) = \exp\left(-\Omega\left(n^{1-\frac{\max\{1,i-1\}}{q}-\frac{i}{2q^2}}\right)\right) = o(1).$$

(Recall that $t = n^{\max\{1,i-1\}/q}$ and the sampling probability is $p = 1/n^{1/2q^2}$.)

Correctness on robust inputs. It remains to show the harder case: when x is a robust input (in our case, $f(x) = 0$), then *all kernel assignments to all daisies* will make the local views that lead to output 1 fail to cross the threshold. This case is harder since, by the asymmetry of N with respect to robust and non-robust inputs, here we need to prove a claim for all kernel assignments to all daisies, whereas in the non-robust case above we only had to argue about the existence of a single assignment to a single kernel.

We begin with a simple observation regarding $\mathcal{D}_0$, then analyse the daisies $\mathcal{D}_i$ for $i > 1$, and deal with the more delicate case of $\mathcal{D}_1$ last. Recall that $\mathcal{D}_0$ is disregarded by the algorithm N, and that by the asymmetry of N with respect to 0- and 1-inputs, this only makes the analysis on robust inputs *easier*. Indeed, N^x is correct when no kernel assignment to any of the $\mathcal{D}_i$'s leads to crossing the threshold τ_i of local views on which the query-based algorithm M outputs 1. Thus, by discarding the query sets in $\mathcal{D}_0$, we only increase the probability of not crossing these thresholds.

Fix $i > 0$ and an arbitrary kernel assignment κ to K_i. Then, the relevant sets that N may sample are those in the support of μ_{x_κ} (recall that x_κ is the word obtained by replacing the bits of x whose coordinates lie in K_i by κ). Since $|K_i| = o(n)$, it follows by the robustness of x that x_κ is ρ-close to x, and thus the weight of the collection $\mathcal{O} \subseteq \mathrm{supp}(\mu_{x_\kappa})$ of query sets that lead to output 1 is at most σ. For the sake of this technical overview, we focus on the worst-case scenario,

where all of these "bad" sets are in the daisy $\mathcal{D}_i$ (i.e., $\mathcal{O} \subseteq \mathcal{D}_i$) and $|\mathcal{O}|$ is as large as possible (i.e., $|\mathcal{O}| = \Theta(n)$), and show that even that will not suffice to cross the threshold τ_i.

By the randomness complexity of the algorithm M, the size of $\mathcal{O}$ is $\Theta(n)$. We apply the Hajnal–Szemerédi theorem and partition $\mathcal{O}$ into $\Theta(t)$ simple daisies of size $\Theta(n/t)$. Recall that the petals of query sets in $\mathcal{D}_i$ have size i and are disjoint; therefore, each of these simple daisies has $\gamma_i \cdot np^i/t$ sampled petals with probability only $\exp\big(-\Omega(np^i/t)\big)$.[20] By an averaging argument, the total number of sampled petals crosses $\tau_i = \gamma_i \cdot np^i$ with probability at most

$$O(t) \cdot \exp\left(-\Omega\left(\frac{np^i}{t}\right)\right) = \exp\left(-\Omega\left(n^{1-\frac{i-1}{q}-\frac{i}{2q^2}}\right)\right) = \exp\left(-\Omega\left(n^{1-\frac{i}{q}+\frac{1}{2q}}\right)\right) ;$$

recall that $t = n^{(i-1)/q}$ and the sampling probability is $p = 1/n^{1/2q^2}$, so $p^i \geq 1/n^{1/2q}$. Since the daisy partition lemma yields a bound of $O(n^{1-i/q})$ for the size of the kernel K_i, a union bound over all $2^{|K_i|}$ kernel assignments ensures the threshold is crossed with probability $o(1)$.[21]

We now analyse $\mathcal{D}_1$ and stress that the need for a separate analysis arises from the looser intersection bound on this daisy: $\mathcal{D}_1$ is a t-daisy with $t = n^{1/q}$, whereas for all other i the bound is $t = n^{(i-1)/q}$. This implies that there is no "gap" between the expected number of queried petals in each simple daisy $\Theta(np/t) = \Theta(n^{1-1/q-1/(2q^2)}) = o(n^{1-1/q})$ and the size of the kernel $|K_1| = O(n^{1-1/q})$, so a union bound as in the case $i > 1$ does not suffice.

This is precisely what the "capping" performed by N on $\mathcal{D}_1$ is designed to address: the query sets $\mathcal{O} \subseteq \mathrm{supp}(\mu_{x_\kappa})$ that lead to output 1 will only be counted by N *if their petals are shared by at most* α *query sets*. Then, by the Hajnal–Szemerédi theorem, we partition $\mathcal{O}$ into $\alpha = \Theta(1)$ simple daisies of size $\Theta(n)$. Each simple daisy will have more than $\tau_i/\alpha = \Theta(np)$ queried petals with probability at most $\exp(-\Omega(np))$, so that the total number of such petals across all simple daisies exceeds τ_j with probability at most $\exp(-\Omega(np))$. This provides the necessary gap: as $\Theta(np) = \Omega(n^{1-1/2q^2})$ and $|K_1| = O(n^{1-1/q})$, a union bound over all $2^{|K_1|}$ assignments to K_1 shows the threshold is crossed with probability $o(1)$.

This concludes our high-level proof of correctness, and thus of Theorem 1 (see Section 6.2 for the full proof). For an overview of how to derive our applications from Theorem 1, see Section 7.

3 Preliminaries

Throughout this paper, constants are denoted by Greek lowercase letters, such as α, β and γ; capital letters of the Latin alphabet generally denote sets (e.g. P and S) or algorithms (e.g., M and N). For each $n \in \mathbb{N}$, we denote by $[n]$ the set $\{1, \ldots, n\}$. Sets S such that $|S| = q$ are called q-sets. The complement of S is denoted $\overline{S}$. We will use Σ to denote an alphabet.

As integrality issues do not substantially change any of our results, equality between an integer and an expression (that may not necessarily evaluate to one) is assumed to be rounded to the nearest integer.

[20] We stress that *the expected number of sampled petals is smaller* in the robust case than in the non-robust one. This is what allows us to show the total number of queried petals is at least $\tau_i = \gamma_i np^i$ with probability $\exp\big(-\Omega(np^i/t)\big)$ in the robust case but $1 - \exp\big(-\Omega(np^i/t)\big)$ in the non-robust, for the same constant γ_i.

[21] Recall that we set the sampling probability to be $p := 1/n^{1/\beta}$ with $\beta = 2q^2$. This choice is justified as follows: the union bound requirement that $2^{|K_i|}$ multiplied by the probability of crossing the threshold be small translates into $1/q - i/\beta > 0$ for all $i \in [q-1]$. Then $i = q-1$ requires $\beta = \Omega(q^2)$.

Multi-sets of sets. To prevent ambiguity, we call (multi-)sets comprised of objects other than points (such as sets, trees or tuples) (multi-)*collections* in this work, and denote them by the calligraphic capitals $\mathcal{D}, \mathcal{S}, \mathcal{T}$.

Distance and proximity. We denote the *absolute distance* between two strings $x, y \in \Sigma^n$ (over the alphabet Σ) by $\bar{\Delta}(x, y) \coloneqq |\{x_i \neq y_i : i \in [n]\}|$ and their *relative distance* (or simply *distance*) by $\Delta(x, y) \coloneqq \frac{\bar{\Delta}(x,y)}{n}$. If $\Delta(x, y) \leq \varepsilon$, we say that x is *ε-close* to y, and otherwise we say that x is *ε-far* from y. The (Hamming) ball of radius ε around x is $B_\varepsilon(x) = \{y \in \Sigma^n : y \text{ is } \varepsilon\text{-close to } x\}$, and the ball $B_\varepsilon(S)$ around a set $S \subseteq \Sigma^n$ is the union over $B_\varepsilon(x)$ for each $x \in S$.

Probability. The uniform distribution over a set S is denoted U_S. We write $X \sim \mu$ to denote a random variable X with distribution μ; the probability of event $[X = s]$ is interchangeably referred to as *weight*, and is denoted $\Pr[X = s]$ (the underlying distribution will be clear from context), and the expected value of X is $\mathbb{E}[X]$. We also write $|\mu|$ as shorthand for $|\text{supp}(\mu)|$, the support size of μ. Below is the version of the Chernoff bound that will be used in this work.

Lemma 3.1 (Chernoff bound)**.** *Let $X_1, \ldots, X_k$ be independent Bernoulli random variables distributed as X. Then, for every $\delta \in [0, 1]$,*

$$\Pr\left[\frac{1}{k}\sum_{i=1}^{k} X_i \geq (1+\delta)\mathbb{E}[X]\right] \leq e^{-\frac{\delta^2 k \mathbb{E}[X]}{3}} \text{ and}$$
$$\Pr\left[\frac{1}{k}\sum_{i=1}^{k} X_i \leq (1-\delta)\mathbb{E}[X]\right] \leq e^{-\frac{\delta^2 k \mathbb{E}[X]}{2}}.$$

Algorithms. We denote by $M^x(z)$ the output of algorithm M given direct access to input z and query access to string x. Probabilistic expressions that involve a randomised algorithm M are taken over the inner randomness of M (e.g., when we write $\Pr[M^x(z) = s]$, the probability is taken over the coin tosses of M). The number of coin tosses M makes is its *randomness complexity*. The maximal number of queries M performs over all strings x and outcomes of its coin tosses is interchangeably referred to as its *query complexity* or *locality* q. When $q = o(n)$, where n is the length of the string x, we say M is a *(q-)local* algorithm. If the queries performed by M are determined in advance (so that no query depends on the result of any other query), M is *non-adaptive*; otherwise, it is *adaptive*. Finally, if M queries each coordinate independently with some probability p, we say it is a *sample-based* algorithm. Since we will want to have an absolute bound on the sample complexity (i.e., the number of coordinates sampled) of sample-based algorithms, we allow them to cap the number of coordinates they sample.

Notation. We will use the following, less standard, notation. An *assignment* to a set S (over alphabet Σ) is a function $a \colon S \to \Sigma$, which may be equivalently seen as a vector in $\Sigma^{|S|}$ whose coordinates correspond to elements of S in increasing order. Its restriction to $P \subseteq S$ is denoted $a_{|P}$. If x is an assignment to S and κ is an assignment to $P \subseteq S$, the *partially replaced assignment* $x_\kappa \in \Sigma^n$ is that which coincides with κ in P and with x in $S \setminus P$ (i.e., $x_{\kappa|P} = \kappa$ and $x_{\kappa|S\setminus P} = x_{S\setminus P}$).

Adaptivity. Adaptive local algorithms are characterised in two equivalent manners: a standard description via decision trees and an alternative that makes more direct use of set systems. Let M be a q-local algorithm for a decision problem (i.e., which outputs an element of $\{0,1\}$) with oracle access to a string over alphabet Σ and no access to additional input (what follows immediately generalises by enumerating over explicit inputs).

The behaviour of M is completely characterised by a collection $\{(T_s, s) : s \in \{0,1\}^r\}$ of *decision trees*, where r is the randomness complexity of M; the trees are $|\Sigma|$-ary, have depth q, edges labeled by elements of Σ, inner nodes labeled by elements of $[n]$ and leaves labeled by elements of $\{0,1\}$. The execution of M^x proceeds as follows. It (1) flips r random coins, obtains a string $s \in \{0,1\}^r$ and chooses the decision tree T_s to execute; (2) beginning at the root, for q steps, queries the coordinate given by the label at the current vertex and follows the edge whose label is the queried value; and (3) outputs the bit given by the label of the leaf thus reached.

Although this offers a complete description of an adaptive algorithm, we choose to use an alternative that is amenable to *daisy lemmas* (see Section 5.1). This is obtained by describing each decision tree with the collection of its branches. More precisely, from $\{(T_s, s) : s \in \{0,1\}^r\}$ we construct $\{(S_{st}, a_{st}, b_{st}, s, t) : s \in \{0,1\}^r, t \in [|\Sigma|^q]\}$, where t identifies which branch the tuple is obtained from. S_{st} is the q-set queried by the t^{th} branch of T_s, while a_{st} is the assignment to S_{st} defined by the edges of this branch and $b_{st} \in \{0,1\}$ is the output at its leaf. We remark that the decision trees may be reconstructed from their branches, so that this is description is indeed equivalent (though we will not need this fact).

We now define the distribution of an algorithm, as well as its distribution under a fixed input.

Definition 3.2 (Induced distribution)**.** *Let M be a q-local algorithm with randomness complexity r described by the collection of decision trees $\{T_s : s \in \{0,1\}^r\}$. The* distribution $\tilde{\mu}^M$ *of M is given by sampling $s \in \{0,1\}^r$ uniformly at random and taking T_s.*

Fix an arbitrary input x to M and, for all $s \in \{0,1\}^r$, let $(S_{st}, x_{|S_{st}}, b_{st}, s, t)$ be the unique tuple defined by the branch of T_s followed on input x. We may thus discard t and the tuple $(S_s, x_{|S_s}, b_s, s)$ is well defined. The distribution μ_x^M is given by sampling $s \in \{0,1\}^r$ uniformly at random and taking the set S_s (the first element of the tuple $(S_s, x_{|S_s}, b_s, s)$).

We note that the contents of the tuple $(S_s, x_{|S_s}, b_s, s)$ describe exactly how M will behave on input x and random string s.

4 Robust local algorithms

We now formally introduce *robust local algorithms*, which capture a wide class of sublinear algorithms, ranging from *property testing* to *locally decodable codes*. Our main result (Theorem 1) holds for any robust local algorithm, and indeed, we obtain our results for coding theory, testing, and proofs of proximity as direct corollaries.

While local algorithms are very well studied, their definition is typically context-dependent, where they are required to perform different tasks (e.g., test, self-correct, decode, perform a local computation) under different promises (e.g., proximity to encoded inputs, being either "close" to or "far" from sets). However, structured promises on the input are (with the exception of degenerate cases) necessary for algorithms that only make a sublinear number of queries. This feature leads naturally to the notion of robustness, which, loosely speaking, a local algorithm satisfies if its output is stable under small perturbations.

In the next subsection, we provide a precise definition of robust local algorithms. Then, in the subsequent subsections, we show how this notion captures property testing, locally testable codes, locally decodable and correctable codes, PCPs of proximity, and other local algorithms.

4.1 Definition

We begin by defining *local algorithms*, which are probabilistic algorithms that receive query access to an input x and explicit parameter z and are required to compute a partial function $f(z,x)$ (which represents a promise problem) by only making a small number of queries to the input x.

Definition 4.1 (Local algorithms)**.** *Let Σ be a finite alphabet, Z a finite set and $\{\mathcal{P}_z : z \in Z\}$ a family of sets $\mathcal{P}_z \subseteq \Sigma^n$ indexed by Z. With $\mathcal{P} \coloneqq \{(z,x) : z \in Z, x \in \mathcal{P}_z\}$, let $f\colon \mathcal{P} \to \{0,1\}$ be a partial function.*[22] *A* q-local algorithm M for computing f *with* error rate σ *receives explicit access to $z \in Z$, query access to $x \in \mathcal{P}_z$, makes at most q queries to x and satisfies*

$$\Pr[M^x(z) = f(z,x)] \geq 1 - \sigma.$$

The parameter q is called the *query complexity* of M, to which we also refer as *locality*. Throughout, when we refer to a *local algorithm*, we mean a q-local algorithm with $q = o(n)$. Another important parameter is the *randomness complexity* of M, defined as the maximal number of coin tosses it makes over all $(z,x) \in Z \times \Sigma^n$ (note that an execution $M^x(z)$ is well-defined even if $x \notin \mathcal{P}_z$).

The following definition formalises the aforementioned natural notion of *robustness*, which is the structural property that underlies local computation.

Definition 4.2 (Robustness)**.** *Let $\rho > 0$. A local algorithm M for computing $f\colon \mathcal{P} \to \{0,1\}$ is* ρ-robust *at the point $(z,x) \in \mathcal{P}$ if $\Pr[M^w(z) = f(z,x)] \geq 1 - \sigma$ for all $w \in B_\rho(x)$. We say that M is* (ρ_0, ρ_1)-robust *if, for all $z \in Z$ and $b \in \{0,1\}$, M is ρ_b-robust at every x such that $f(z,x) = b$.*

If a local algorithm M is (ρ_0, ρ_1)-robust and $\max\{\rho_0, \rho_1\} = \Omega(1)$ (a constant independent of n), we simply call M *robust*. Note that non-trivial robustness is only possible because f is a partial function; that is, the local algorithm M solves a *promise problem* where, for every parameter z, the algorithm is promised to receive an input from $\mathcal{P}_{z,0} \coloneqq f^{-1}(z,0)$ on which it should output 0, or an input from $\mathcal{P}_{z,1} \coloneqq f^{-1}(z,1)$ on which it should output 1.

Remark 4.3 (One-sided robustness)**.** For our main result (Theorem 1), it suffices to have *one-sided robustness*, i.e., (ρ_0, ρ_1)-robustness where only one of ρ_0, ρ_1 is non-zero. For example, in the setting of property testing with proximity parameter ε we only have $(\varepsilon, 0)$-robustness (see Section 4.2 for details). To simplify notation, we refer to $(\rho, 0)$-robust local algorithms as ρ-robust.

Remark 4.4 (Larger alphabets)**.** The definition of local algorithms can be further generalised to a constant-size output alphabet Γ, in which case the partial function is $f\colon \Sigma^n \to \Gamma$; we assume $\Gamma = \{0,1\}$ for simplicity of exposition, but note that our results extend to larger output alphabets in a straightforward manner.

We proceed to show how to capture various well-studied families of sublinear algorithms (such as testers, local decoders, and PCPs) using the notion of robust local algorithms.

[22] We remark that allowing only rectangles $\mathcal{P} = Z \times \mathcal{Q}$ as the domain of f suffices for most of our applications (e.g., testers and local decoders), but not all. For example, in a MAP for a property Π, there may be inputs $x \in \Pi$ that are only contained in $\mathcal{P}_z$ for a single $z \in Z$. (See Section 7.3.)

4.2 Property testing

Property testing [RS96, GGR98] deals with probabilistic algorithms that solve approximate decision problems by making a small number of queries to their input. More specifically, a tester is required to decide whether its input is in a set Π (i.e., has the property Π) or whether it is ε-far from any input in Π.

Definition 4.5 (Testers)**.** *An* ε-tester *with* error rate σ *for a* property $\Pi \subseteq \Sigma^n$ *is a probabilistic algorithm* T *that receives query access to a string* $x \in \Sigma^n$*. The tester* T *performs at most* $q = q(\varepsilon, n)$ *queries to* x *and satisfies the following two conditions.*

1. *If* $x \in \Pi$*, then* $\Pr[T^x = 1] \geq 1 - \sigma$*.*
2. *For every* x *that is* ε*-far from* Π *(i.e.,* $x \in \overline{B_\varepsilon(\Pi)}$*), then* $\Pr[T^x = 0] \geq 1 - \sigma$*.*

We are interested in the regime where $\varepsilon = \Omega(1)$ (i.e., ε is a fixed constant independent of n), and assume it to be the case in the remainder of this discussion.

Note that testers are *not* robust with respect to inputs in the property Π, as changing a single coordinate of an input $x \in \Pi$ could potentially lead to an input outside Π. Moreover, an ε-tester does not immediately satisfy one-sided robustness, as inputs that are on the boundary of the ε-neighbourhood of Π are not robust (see figure Fig. 1b).

However, by increasing the value of the proximity parameter by a factor of 2, we can guarantee that every point that is 2ε-far from Π satisfies the robustness condition. The following claim formalises this statement and shows that testers can be cast as robust local algorithms.

Claim 4.6. *An* ε*-tester* T *for property* $\Pi \subseteq \Sigma^n$ *is an* $(\varepsilon, 0)$*-robust local algorithm, with the same parameters, for computing the function* f *defined as follows.*

$$f(x) = \begin{cases} 1, & \text{if } x \in \Pi \\ 0, & \text{if } x \text{ is } 2\varepsilon\text{-far from } \Pi. \end{cases}$$

Proof. By definition, the tester T is a local algorithm for computing f; denote its error rate by σ. We show it satisfies (one-sided) robustness with respect to f. Let $x \in \Sigma^n$ be an input that is 2ε-far from Π, and consider $y \in B_\varepsilon(x)$. By the triangle inequality, we have that y is ε-far from Π. Thus, $\Pr[T^y = 0] \geq 1 - \sigma$, and so T is an $(\varepsilon, 0)$-robust local algorithm for f. □

Remark 4.7 (Robustness vs proximity tradeoff)**.** The notion of a tester with proximity parameter ε and that of an ε-robust tester with proximity parameter 2ε coincide. Moreover, there is a tradeoff between the size of the promise captured by the partial function f and the robustness parameter ρ: taking any $\varepsilon' > \varepsilon$, the tester T is a ρ-robust local algorithm with $\rho = \varepsilon' - \varepsilon$ for computing the function

$$f(x) = \begin{cases} 1, & \text{if } x \in \Pi \\ 0, & \text{if } x \text{ is } \varepsilon'\text{-far from } \Pi. \end{cases}$$

As ε' increases, the robustness parameter ρ increases and the size of the domain of definition of f decreases. In particular, taking $\varepsilon' = \varepsilon$ makes T a $(0, 0)$-robust algorithm (i.e., an algorithm that is not robust).

4.3 Local codes

We consider error-correcting codes that admit local algorithms for various tasks, such as testing, decoding, correcting, and computing functions of the message. Recall that a *code* $C\colon \{0,1\}^k \to \{0,1\}^n$ is an injective mapping from *messages* of length k to codewords of *blocklength* n. The *rate* of the code C is defined as k/n. The *relative distance* of the code is the minimum, over all distinct messages $x, y \in \Sigma^k$, of $\Delta(C(x), C(y))$. We shall sometimes slightly abuse notation and use C to denote the set of all of its codewords $\{C(x) : x \in \Sigma^k\} \subset \Sigma^n$. Note that we focus on *binary* codes, but remind that the extension to larger alphabets is straightforward. In the following, we show how to cast the prominent notions of local codes as robust local algorithms.

4.3.1 Locally testable codes

Locally testable codes (LTCs) [GS06] are codes that admit algorithms that distinguish codewords from strings that are far from being valid codewords, using a small number of queries.

Definition 4.8 (Locally Testable Codes (LTCs))**.** *A code $C\colon \{0,1\}^k \to \{0,1\}^n$ is locally testable, with respect to proximity parameter ε and error rate σ, if there exists a probabilistic algorithm T that makes q queries to a purported codeword w such that:*

1. *If $w = C(x)$ for some $x \in \{0,1\}^k$, then $\Pr[T^w = 1] \geq 1 - \sigma$.*
2. *For every w that is ε-far from C, we have $\Pr[T^x = 0] \geq 1 - \sigma$.*

Note that the algorithm T that an LTC admits is simply an ε-tester for the property of being a valid codeword of C. Thus, by Claim 4.6, we can directly cast T as a robust local algorithm.

4.3.2 Locally decodable codes

Locally decodable codes (LDCs) [KT00] are codes that admit algorithms for decoding each individual bit of the message of a moderately corrupted codeword by only making a small number of queries to it.

Definition 4.9 (Locally Decodable Codes (LDCs))**.** *A code $C\colon \{0,1\}^k \to \{0,1\}^n$ is locally decodable with* decoding radius *δ and error rate σ if there exists a probabilistic algorithm D that, given index $i \in [k]$, makes q queries to a string w promised to be δ-close to a codeword $C(x)$, and satisfies*

$$\Pr[D^w(i) = x_i] \geq 1 - \sigma.$$

Note that local decoders are significantly different from local testers and testing in general. Firstly, decoders are given a promise that their input is *close* to a valid codeword (whereas testers are promised to either receive a perfectly valid input, or one that is far from being valid). Secondly, a decoder is given an index as an explicit parameter and is required to perform a different task (decode a different bit) for each parameter (see Fig. 1a).

Nevertheless, local decoders can also be cast as robust local algorithms. In fact, unlike testers, they satisfy *two-sided* robustness (i.e., both 0-inputs and 1-inputs are robust). In the following, note that since inputs near the boundary of the decoding radius are *not* robust, we reduce the decoding radius by a factor of 2.

Claim 4.10. *A local decoder D with decoding radius δ for the code $C\colon \{0,1\}^k \to \{0,1\}^n$ is a $(\delta/2, \delta/2)$-robust local algorithm for computing the function f defined as follows.*

$$f(z, w) = x_z, \text{ if } x \in \{0,1\}^k \text{ is such that } w \text{ is } \delta/2\text{-close to } C(x).$$

Proof. Take any $w \in \{0,1\}^n$ that is $\delta/2$-close to $C(x)$. Then, (w, z) is in the domain of definition of f for all explicit inputs $z \in [k]$. Now let $w' \in B_{\delta/2}(w)$ and note that w' is still within the decoding radius of D. Hence, the decoder D outputs x_z with probability $1 - \sigma$, as required. Moreover, this holds regardless whether $x_z = 0$ or $x_z = 1$, and so D is $(\delta/2, \delta/2)$-robust. □

Remark 4.11 (Robustness vs decoding radius tradeoff)**.** A local decoder has decoding radius δ if and only if it is $\delta/2$-robust with decoding radius $\delta/2$, and a tradeoff between promise size and robustness parameter likewise holds in this case: for any $\delta' < \delta$, the decoder D is a $(\delta - \delta', \delta - \delta')$-robust algorithm for the restriction of f to the δ'-neighbourhood of the code C. In particular, D is a (δ, δ)-robust algorithm with the domain of f defined to be the code C.

4.3.3 Relaxed locally decodable codes

Relaxed locally decodable codes (relaxed LDCs) [BGH+06] are codes that admit a natural relaxation of the notion of local decoding, in which the decoder is allowed to output a special abort symbol $\perp$ on a small fraction of indices, indicating it detected an inconsistency, but never erring with high probability.

Definition 4.12 (Relaxed LDCs)**.** *A code $C\colon \{0,1\}^k \to \{0,1\}^n$ whose distance is δ_C is a q-local relaxed LDC with success rate ρ, decoding radius $\delta \in (0, \delta_C/2)$ and error rate $\sigma \in (0, 1/3]$ if there exists a randomised algorithm D, known as a* relaxed decoder*, that, on input $i \in [k]$, makes at most q queries to an oracle w and satisfies the following conditions.*

1. *Completeness: For any $i \in [k]$ and $w = C(x)$, where $x \in \{0,1\}^k$,*

$$\Pr[D^w(i) = x_i] \geq 1 - \sigma \, .$$

2. *Relaxed Decoding: For any $i \in [k]$ and $w \in \{0,1\}^n$ that is δ-close to a (unique) codeword $C(x)$,*

$$\Pr[D^w(i) \in \{x_i, \perp\}] \geq 1 - \sigma \, .$$

3. *Success Rate: There exists a constant $\xi > 0$ such that, for any $w \in \{0,1\}^n$ that is δ-close to a codeword $C(x)$, there exists a set $I_w \subseteq [k]$ of size at least ξk such that for every $i \in I_w$,*

$$\Pr[D^w(i) = x_i] \geq 2/3 \, .$$

Note that strictly speaking, the special abort symbol makes it so that relaxed local decoders do not fully fit Definition 4.1, as the input-output mapping f becomes one-to-many. Nevertheless, a simple generalisation of local algorithms, which allows an additional abort symbol, enables us to capture relaxed LDCs as robust local algorithms as well. We show this in Section 7.2.

4.3.4 Locally correctable codes

The notion of locally correctable codes (LCCs) is closely related to that of LDCs, except that rather than admitting an algorithm that can decode any individual *message* bit, LCCs admit an algorithm that can correct any corrupted *codeword* bit of a moderately corrupted codeword.

Definition 4.13 (Locally Correctable Codes (LCCs))**.** *A code $C\colon \{0,1\}^k \to \{0,1\}^n$ is locally correctable with* correcting radius *δ and error rate σ if there exists a probabilistic algorithm D that, given index $j \in [n]$, makes q queries to a string w promised to be δ-close to a codeword $C(x)$ and satisfies*

$$\Pr[D^w(j) = C(x)_j] \geq 1 - \sigma.$$

A straightforward adaptation of Claim 4.10 yields the following claim.

Claim 4.14. *A local corrector D with correcting radius δ for the code $C\colon \{0,1\}^k \to \{0,1\}^n$ is a $(\delta/2, \delta/2)$-robust local algorithm for computing the function f defined as follows.*

$$f(z, w) = C(x)_z, \text{ if } x \in \{0,1\}^k \text{ is such that } w \text{ is } \delta/2\text{-close to } C(x).$$

4.3.5 Universal locally testable codes

Universal locally testable codes (universal LTCs) [GG18] are codes that admit local tests for membership in numerous possible subcodes, allowing for testing properties of the encoded message.

Definition 4.15 (Universal LTCs)**.** *A universal LTC $C\colon \{0,1\}^k \to \{0,1\}^n$ for a family of functions $\mathcal{F} = \left\{f_i : \{0,1\}^k \to \{0,1\}\right\}_{i\in[M]}$ is a code such that for every $i \in [M]$ the subcode $\{C(x) : f_i(x) = 1\}$ is locally testable.*

Note that ULTCs trivially generalise LTCs, as well as generalise relaxed LDCs (see details in [GG18, Appendix A]). Since universal testers can be viewed as algorithms that receive an explicit parameter $i \in [M]$ and invoke an ε-tester for the property $\{C(x) : f_i(x) = 1\}$, then by applying Claim 4.6 to each value of the parameter i they can be cast as robust local algorithms.

4.4 PCPs of proximity

PCPs of proximity (PCPPs) [BGH+06] are probabilistically checkable proofs wherein the verifier is given query access not only to the proof, but also to the input. The PCPP verifier is then required to probabilistically check whether the statement is correct by only making a constant number of queries to both input and proof.

Definition 4.16. *A PCP of proximity (PCPP) for a language L with proximity parameter ε, error rate σ and query complexity q, consists of a probabilistic algorithm V, called the verifier, that receives query access to* both *an input $x \in \Sigma^n$ and a proof $\pi \in \{0,1\}^m$. The verifier V is allowed to make q queries to (x, π) and satisfies the following:*

1. *for every $x \in L$ there exists a proof π such that $\Pr\left[V^{(x,\pi)} = 1\right] \geq 1 - \sigma$; and*
2. *for every x that is ε-far from L and every proof π, it holds that $\Pr\left[V^{(x,\pi)} = 0\right] \geq 1 - \sigma$.*

We observe that PCPs of proximity with canonical proofs [GS06] (i.e., such that the verifier rejects statement-proof pairs that are far from being the concatenation of a valid statement with a valid proof for it) admit verifiers that are robust local algorithms. Using the tools of [DGG19], who show that PCPPs can be endowed with the canonicity property at the cost of polynomial blowup in proof length, we can obtain robust local algorithms for general PCPPs.

Claim 4.17. *A PCPP for a language $L \subseteq \Sigma^n$ with proximity parameter $\varepsilon > 0$, error rate σ and query complexity q can be transformed into a PCPP for L with proximity parameter 2ε, whose verifier is an $(\varepsilon, 0)$-robust local algorithm with the same query complexity and error rate.*

Sketch of proof. Let V be a PCPP verifier with proximity parameter ε and error rate σ for $L \subseteq \Sigma^n$, that makes at most q queries to its input-proof pair $(x, \pi) \in \Sigma^n \times \{0,1\}^m$. By [DGG19, Section 3], there exists a PCPP verifier V' for L with $\mathrm{poly}(m)$ proof length (as well as proximity parameter ε, error rate σ and query complexity q) that satisfies the following strengthening of the conditions in Definition 4.16: there is a set of *canonical proofs* $\{\pi_x\}_{x \in L}$ such that

1. for every $x \in L$, it holds that $\Pr\left[V^{(x,\pi_x)} = 1\right] \geq 1 - \sigma$; and
2. for every (x, π) that is ε-far from (y, π_y) for all $y \in L$, it holds that $\Pr\left[V^{(x,\pi)} = 0\right] \geq 1 - \sigma$.

In other words, V' is an ε-tester for the property $\Pi \coloneqq \{(x, \pi_x) : x \in L\}$, and we invoke Claim 4.6. □

Non-interactive proofs of proximity. MA proofs of proximity (MAPs) [GR18, FGL14] are proof systems that can be viewed as a property testing analogue of NP proofs. The setting of MAPs is very similar to that of PCPPs, with the distinction that the purported proof is of sublinear size and is given explicitly, i.e., the MAP verifier can read the entire proof. We remark that an equivalent description of a MAP as a covering by partial testers [FGL14] is used in this work, so that *every fixed proof string* defines a tester, and Claim 4.6 applies. We cover this in Section 7.3.

5 Technical lemmas

In the section we provide an arsenal of technical tools for analysing robust local algorithms, which we will then use to prove our main result in Section 6. The order in which we present the tools is according to their importance, starting with the most central lemmas.

Specifically, in Section 5.1 we discuss the notion of relaxed sunflowers that we shall need, called daisies, then state and prove a daisy partition lemma for multi-collections of sets. In Section 5.2, we apply the Hajnal-Szemerédi theorem to derive a sampling lemma for daisies. In Section 5.3, we prove a simple yet vital volume lemma for robust local algorithms, which will be used throughout our analysis. Finally, in Section 5.4 we adapt generic transformations (amplification and randomness reduction) to our setting of robust local algorithms.

5.1 Relaxed sunflowers

We discuss the central technical tool used in the transformation to sample-based algorithms, which is a relaxation of combinatorial sunflowers, referred to as *daisies* [FLV15, GL21]. We extend the definition of daisies to multi-sets, then state and prove the particular variant of a daisy lemma that we shall need.

Definition 5.1 (Daisy)**.** *Suppose $\mathcal{S}$ is a multi-collection of subsets of $[n]$ (i.e., subsets may repeat). $\mathcal{S}$ is an h-daisy (where $h\colon \mathbb{N} \to \mathbb{N}$) with petals of size j and* kernel *$K \subseteq [n]$ if the following holds: every $S \in \mathcal{S}$ has a* petal *$S \setminus K$ with $|S \setminus K| = j$ and, for every $k \in [j]$, there exists a subset $P_k \subseteq S \setminus K$ with $|P_k| \geq k$ whose elements are contained in at most $h(k)$ sets from $\mathcal{S}$.*
A daisy with pairwise disjoint petals (1-daisy) is referred to as a simple daisy*.*

We remark that the notion of a daisy relaxes the standard definition of a sunflower in two ways: (1) the kernel is not required to equal the pairwise intersection of all sets in the collection, its structure is unconstrained; and (2) the *petals* $\mathcal{P} = \{S \setminus K : S \in \mathcal{D}\}$ need not be pairwise disjoint, but rather, each point outside of the kernel can be contained in at most $h(j)$ sets of $\mathcal{D}$; see Fig. 2b. Note that Definition 5.1, in contrast to sunflowers (for which pairwise disjointness disallows multiple copies of a same set), applies to *multi-sets*.

These relaxations, unlike in the case of sunflowers, allow us to arbitrarily partition any collection of subsets into a collection of daisies with strong structural properties, as Lemma 5.2 shows.

Lemma 5.2 (Daisy partition lemma for multi-collections)**.** *Let $\mathcal{S}$ be a multi-collection of q-sets of $[n]$, and define the function $h\colon \mathbb{N} \to \mathbb{N}$ as follows:*

$$h(k) = n^{\frac{\max\{1,k-1\}}{q}}.$$

Then, there exists a collection $\{\mathcal{D}_j : 0 \leq j \leq q\}$ such that

1. *$\{\mathcal{D}_j\}$ is a partition of $\mathcal{S}$, i.e., $\bigcup_{j=0}^{q} \mathcal{D}_j = \mathcal{S}$ and $\mathcal{D}_j \cap \mathcal{D}_k = \emptyset$ when $j \neq k$.*
2. *For every $0 \leq j \leq q$, there exists a set $K_j \subseteq [n]$ of size $|K_j| \leq q|\mathcal{S}| \cdot n^{-\max\{1,j\}/q}$ such that $\mathcal{D}_j$ is an h-daisy with kernel K_j and petals of size j. Moreover, the kernels form an incidence chain $\varnothing = K_q \subseteq K_{q-1} \subseteq \cdots \subseteq K_1 = K_0$.*

Proof. We construct the collections $\{\mathcal{D}_j : 0 \leq j \leq q\}$ in a greedy iterative manner, as follows.

1. Define $\mathcal{S}_0 \coloneqq \mathcal{S}$.
2. Inductively define, for each $0 \leq j \leq q-1$:
   (a) *Kernel construction:* Define K_j as the set of points in $[n]$ that are contained in at least $h(j+1)$ sets from $\mathcal{S}$.
   (b) *Daisy construction:* Set $\mathcal{D}_j$ to be all the sets $S \in \mathcal{S}_j$ such that $|S \setminus K_j| = j$.
   (c) Set $\mathcal{S}_{j+1}$ to be $\mathcal{S}_j \setminus \mathcal{D}_j$.
3. Finally, set $\mathcal{D}_q = \mathcal{S}_q$ and $K_q = \emptyset$.

We now prove that this construction yields daisies with the required properties. For ease of notation, let d_i be the number of sets in $\mathcal{S}$ containing i for each $i \in [n]$. By definition, $\mathcal{S}_q \subseteq \mathcal{S}_{q-1} \subseteq \cdots \subseteq \mathcal{S}_0$ and $\mathcal{D}_j \subseteq \mathcal{S}_j$ for all j. Since $\mathcal{D}_{j-1} \cap \mathcal{S}_j = \varnothing$ for all $j \in [q]$, it follows that $\mathcal{D}_j \cap \mathcal{D}_k = \varnothing$ when $j \neq k$. Also, since $\mathcal{S} = \mathcal{S}_q \cup \bigcup_{0 \leq j < q} \mathcal{D}_j$ and $\mathcal{S}_q = \mathcal{D}_q$, we have $\mathcal{S} = \bigcup_{0 \leq j \leq q} \mathcal{D}_j$.

Since $\mathcal{S}$ is comprised of q-sets,

$$\sum_{i \in [n]} d_i = q|\mathcal{S}|.$$

By the kernel construction, for each $j \in \{0, 1, \ldots, q\}$, K_j is the set of all $i \in [n]$ such that $d_i \geq h(j+1)$, which implies $\sum_{i \in K_j} d_i \geq |K_j| \cdot h(j+1)$. Therefore,

$$q|\mathcal{S}| = \sum_{i \in [n]} d_i \geq \sum_{i \in K_j} d_i \geq |K_j| \cdot h(j+1)$$

and thus $|K_j| \leq q|\mathcal{S}|/h(j+1) = q|\mathcal{S}| \cdot n^{-\max\{1,j\}/q}$. Note, also, that the kernel construction ensures not only $K_j \subseteq K_{j-1}$ when $j \in [q]$, but also $K_0 = K_1$ because $h(1) = h(2)$.

Since the petals of each $S \in \mathcal{D}_j$ have size exactly j by construction, all that remains to be proven is the intersection condition on the petals that makes $\mathcal{D}_j$ an h-daisy; namely, that for every $k \in [j]$, there exists a subset $P_k \subseteq S \setminus K$ with $|P_k| \geq k$ whose elements are contained in at most $h(k)$ sets from $\mathcal{D}_j$. Assume for the sake of contradiction that this condition does not hold.

Let $j \in [q]$ be the smallest value for which $\mathcal{D}_j$ is not an h-daisy and $S \in \mathcal{D}_j$ be a set that violates the intersection condition; then take $k \leq j$ to be the smallest subset size such that every $P_k \subseteq S \setminus K_j$ with size k has an element $i \in P_k$ with $d_i > h(k)$ (equivalently said, j and k are minimal such that the subset $L \subset S \setminus K_j$, comprised of all i with $d_i \leq h(k)$, has size at most $k-1$).

Suppose first that $k = 1$. Then, every $i \in S \setminus K_j$ is such that $d_i > h(2) = h(1)$ and thus $S \setminus K_j \subseteq K_0$. But this implies $S \in \mathcal{D}_0$ (since $|S \setminus K_0| = 0$), a contradiction because the intersection condition holds by vacuity on empty petals.

Suppose now that $k > 1$. The subset

$$L \coloneqq \{i \in S \setminus K_j : d_i \leq h(k)\}$$

contains at most $k-1$ points; however, by minimality of k, at least $k-1$ distinct points $i \in L$ satisfy $d_i \leq h(k-1) \leq h(k)$. Therefore, $|L| = k-1$ and

$$L = \{i \in S \setminus K_j : d_i \leq h(k-1)\} \ .$$

By the definition of L, every $i \in S \setminus (K_j \cup L)$ satisfies $d_i > h(k)$, so that $i \in K_{k-1}$; therefore, $S \subseteq K_{k-1} \cup K_j \cup L$. Since the kernels form an incidence chain, $K_j \subseteq K_{k-1}$ and thus $S \setminus K_{k-1} = L$. But then $|S \setminus K_{k-1}| = |L| = k-1$, so that $S \in \mathcal{D}_{k-1}$, contradicting the fact that $S \in \mathcal{D}_j$ (because $k-1 < j$ and $\{\mathcal{D}_j\}$ is a partition). □

The following claim shows an upper bound on the total number of sets in an h-daisy that may intersect a given petal. It will be useful in order to partition a daisy into *simple* daisies, as the next section will show.

Claim 5.3. *Let $\mathcal{S}$ be a multi-collection of q-sets and $\{\mathcal{D}_j : 0 \leq j \leq q\}$ be a daisy partition obtained by an application of Lemma 5.2. Then, for every $j \in [q]$ and $S \in \mathcal{D}_j$, the number of sets in $\mathcal{D}_j$ whose petals intersect $S \setminus K_j$ (including S itself) is at most $2h(j) = 2n^{\max\{1,j-1\}/q}$.*

Proof. Let S be an arbitrary set in $\mathcal{D}_j$. We name the elements in $S \setminus K_j$ by $u_1, u_2, \ldots, u_j$ (by Lemma 5.2, every $S \in \mathcal{D}_j$ satisfies $|S \setminus K_j| = j$). For every $k \in [j]$, let d_k be the number of sets of $\mathcal{D}_j$ that u_k is a member of.

Assume without loss of generality that $d_k \leq d_\ell$ for every k and ℓ in $[j]$ such that $k < \ell$, as otherwise we can rename $u_1, u_2, \ldots, u_j$ so that this holds.

By the definition of an h-daisy, for every $\ell \in [j]$, there exists a set of ℓ elements $k \in [j]$ that satisfy $d_k \leq h(\ell)$. Thus, $[\ell]$ is such a set and we know that $d_\ell \leq h(\ell)$. As the number of petals that intersect $S \setminus K_j$ is at most $\sum_{k=1}^{j} d_k$, we get that

$$\sum_{k=1}^{j} d_k \leq \sum_{k=1}^{j} h(k) = \sum_{k=1}^{j} n^{\frac{\max\{1,k-1\}}{q}} \leq 2h(j).$$

The last equality follows directly from the value of $h(k)$. □

5.2 Sampling daisies and the Hajnal-Szemerédi theorem

Concentration of measure is an essential ingredient in our proofs, which we first illustrate via a simplified example. Consider a collection of singletons that comprise the petals of a combinatorial sunflower: sets $P_1, \dots P_k$, all disjoint and of size 1, contained in the ground set $[n]$. If we perform binomial sampling of the ground set (sampling each $i \in [n]$ independently with probability p), the Chernoff bound ensures that the number of sampled petals is close to its expectation. Defining X_i as the random variable that indicates whether P_i was sampled, we have lower and upper tail bounds that guarantee the number of queried petals is concentrated around pn except with exponentially small probability. Note, too, that the same holds for larger petals: if P_i is a j-set for all i, the number of queried petals is concentrated around $p^j n$.

Now consider the case where $P_1, \dots, P_k$ are petals of a *daisy*. In this case the Chernoff bound does not apply, since the indicator random variables X_i are no longer independent; however, the structure of a daisy ensures there is not too much intersection among these petals, which gives means to control the correlation between these random variables. It is thus reasonable to expect that *sampling a daisy is similar to sampling a sunflower*. This intuition is formalised by making use of the Hajnal-Szemerédi theorem [HS70], which we state next.

Theorem 5.4. *Let G be a graph with m vertices and maximum degree Δ. Then, for any $k \geq \Delta + 1$, there exists a k-colouring of the vertices of G such that every colour class has size either $\lfloor m/k \rfloor$ or $\lceil m/k \rceil$.*

We remind that integrality does not cause issues in our analyses, and we thus assume all colour classes have size n/k. By encoding the sets of a daisy as the vertices of an "intersection graph", the fact that petals have bounded intersection translates into a graph with bounded maximum degree. Applying the Hajnal-Szemerédi theorem to this graph, we are able to partition the original daisy into a small number of large *simple* daisies.

Lemma 5.5. *A daisy $\mathcal{D}$ with kernel K, such that each one of its petals has a non-empty intersection with at most $t-1$ other petals, can be partitioned into t simple daisies with the same kernel, each of size $|\mathcal{D}|/t$.*

Proof. Construct a graph G with vertex set $\mathcal{D}$ by placing an edge between vertices S and S' when $(S \cap S') \setminus K \neq \varnothing$. By definition, the maximum degree of G is $\Delta(G) \leq t-1$. The Hajnal-Szemerédi theorem implies G is colourable with t colours, where each colour class has size $|G|/t$. This partition of the vertex set corresponds to a partition of the daisy $\mathcal{D}$ into simple daisies $\{\mathcal{S}_j : j \in [t]\}$, each of size $|\mathcal{D}|/t$. □

5.3 The volume lemma

This section proves a key lemma that makes use of daisies to prove a certain structure on the sets that a *robust* local algorithm may query. Loosely speaking, the volume lemma ensures that in order for a collection of sets to be queried with high enough probability, it must cover a sufficiently large fraction of the input's coordinates.

Let M be a q-local algorithm that computes a partial function f with error rate σ (we assume the explicit input to be fixed and omit it). Recall that, for each input $x \in \Sigma^n$, the algorithm M queries according to a distribution μ_x over a multi-collection of q-sets, as defined in Definition 3.2.

Lemma 5.6 (Volume lemma)**.** *Fix $x \in \Sigma^n$ in the domain of f. If there exists a ρ-robust $y \in \Sigma^n$ such that $f(y) \neq f(x)$, then every collection $\mathcal{S} \subseteq \operatorname{supp}(\mu_x)$ such that $|\cup\mathcal{S}| = |\cup_{S\in\mathcal{S}} S| < \rho n$ satisfies $\mu_x(\mathcal{S}) \leq 2\sigma$.*

Proof. Suppose, by way of contradiction, that there exists $\mathcal{S} \subseteq \operatorname{supp}(\mu_x)$ such that $\mu_x(\mathcal{S}) > 2\sigma$ and $|\cup\mathcal{S}| < \rho n$.

For notational simplicity, assume without loss of generality that $f(x) = 1$, and take a ρ-robust $y \in \Sigma^n$ such that $f(y) = 0$. Define w to match x in the coordinates covered by $\cup\mathcal{S}$, and to match y otherwise. Then w is ρ-close to y, so that M outputs 0 when its input is w with probability at least $1 - \sigma$.

When the algorithm samples a decision tree that makes it query $S \in \mathcal{S}$, then M behaves *exactly* as it would on input x, which happens with probability at least $\mu_x(\mathcal{S}) > 2\sigma$. But the algorithm outputs 1 on input x with probability at most σ, and thus outputs 0 on input w with probability greater than σ, in contradiction with the robustness of y. □

Remark 5.7. The volume lemma requires an arbitrary ρ-robust y with $f(y) \neq f(x)$. It thus suffices that *a single* such ρ-robust point exists for the volume lemma to hold *for every x' such that $f(x') = f(x)$*.

5.4 Generic transformations

This section provides two standard transformations that improve parameters of an algorithm: error reduction (Claim 5.8) and randomness reduction (Claim 5.9), which, applied in conjunction, imply Lemma 5.10. These apply generally to randomised algorithms for decision problems, and, when applied to robust local algorithms, *both transformations compute the same function and preserve robustness*. We defer the proofs in this section to Appendix A.

The following claim is an adaptation of a basic fact regarding randomised algorithms: performing independent runs and selecting the output via a majority rule decreases the error probability exponentially.

Claim 5.8 (Error reduction)**.** *Let M be a (ρ_0, ρ_1)-robust algorithm for $f\colon \mathcal{P} \to \{0,1\}$ (where $\mathcal{P} \subset Z \times \Sigma^n$) with error rate $\sigma \leq 1/3$, query complexity q and randomness complexity r.*

For any $\sigma' > 0$, there exists a (ρ_0, ρ_1)-robust algorithm N for computing the same function with error rate σ', query complexity $O(q \log(1/\sigma')/\sigma)$ and randomness complexity $O(r \log(1/\sigma')/\sigma)$.

Next, we state a transformation that yields an algorithm with twice the error rate and significantly reduced randomness complexity. This, in turn, provides an upper bound on the number of q-sets queried by the algorithm, such that an application of Lemma 5.2 to this multi-collection yields daisies

with kernels of sublinear size. Such a bound on the size of the kernels is crucial to ensure correctness of the sample-based algorithm we construct in Section 6.1. Our proof adapts the technique of Goldreich and Sheffet [GS10], which in turn builds on the work of Newman [New91].

Claim 5.9 (Randomness reduction)**.** *Let M be a (ρ_0, ρ_1)-robust algorithm for $f\colon \mathcal{P} \to \{0,1\}$ (where $\mathcal{P} \subset Z \times \Sigma^n$) with error rate σ, query complexity q and randomness complexity r.*

There exists a (ρ_0, ρ_1)-robust algorithm N for computing the same function with error rate 2σ and query complexity q, whose distribution $\tilde{\mu}^N$ has support size $3n \ln |\Sigma|/\sigma$. In particular, the randomness complexity of N is bounded by $\log(n/\sigma) + \log\log|\Sigma| + 2$.

In the next section, we need a combination of error reduction and randomness reduction, which the following lemma provides.

Lemma 5.10. *Assume there exists a ρ-robust algorithm M for computing f with query complexity ℓ, error rate $1/3$ and arbitrary randomness complexity. Then there exists a ρ-robust q-local algorithm M' for f with error rate*

$$\sigma = \frac{1}{4q}$$

such that $\frac{q}{\log 8q} = O(\ell)$, or, equivalently,

$$q = O(\ell \log \ell).$$

Moreover, the distribution of M' is uniform over a multi-collection of decision trees of size $6n \ln |\Sigma|/\sigma$.

6 Proof of Theorem 1

This section contains the main technical contribution of our work: a proof that every robust local algorithm with query complexity q can be transformed into a *sample-based* local algorithm with sample complexity $n^{1-1/O(q^2 \log^2 q)}$. We begin by providing a precise statement of Theorem 1. In the following, we remind that when the error rate of an algorithm is not stated, it is assumed to be $1/3$.

Theorem 6.1 (Theorem 1, restated)**.** *Suppose there exists a (ρ_0, ρ_1)-robust local algorithm M for the function $f\colon \mathcal{P} \to \{0,1\}$ (where $\mathcal{P} \subset Z \times \Sigma^n$) with query complexity ℓ and $\max\{\rho_0, \rho_1\} = \Omega(1)$. Then, there exists a* sample-based *algorithm N for f with sample complexity $\gamma \cdot n^{1-1/2q^2}$, where $q = O(\ell \log \ell)$ and $\gamma = O(|\Sigma|^q \ln |\Sigma|)$.*

Note that when $q = \Omega(\sqrt{\log n})$ or $|\Sigma|^q = \Omega\big(n^{1/2q^2}\big)$, the algorithm we obtain samples linearly many coordinates, and the statement becomes trivial. Therefore, hereafter we assume that the query complexity of M satisfies $\ell \le \sqrt[5]{\log n}$ (so $q = \Theta(\sqrt[5]{\log n} \log\log n) = o(\sqrt{\log n})$) and the alphabet size satisfies $|\Sigma| \le 2^{\sqrt[6]{\log n}}$ (so $|\Sigma|^q \le n^{1/q^3}$).

We proceed to prove Theorem 6.1. Specifically, in Section 6.1 we construct a sample-based local algorithm N from the (ρ_0, ρ_1)-robust local algorithm M in the hypothesis of Theorem 6.1; in Section 6.2, we analyse our sample-based algorithm N; and in Section 6.5 we conclude the proof by showing the lemmas proved throughout the analysis indeed imply correctness of N.

6.1 Construction

Hereafter, let $f\colon \mathcal{P} \to \{0,1\}$ be the function in the hypothesis of Theorem 6.1. As the following treatment is the same for all explicit inputs $z \in Z$, we assume it to be fixed and omit it from the notation. We also assume without loss of generality that ρ_0 is a constant strictly greater than 0 (if this is not the case we simply exchange the 0 and 1 values in the truth table of f). We set $\rho = \rho_0$.

Let M be the algorithm in the hypothesis of Theorem 6.1. We apply Lemma 5.10 and obtain a (ρ_0, ρ_1)-robust local algorithm M' for the same problem, with query complexity $q = O(\ell \log \ell)$ and error rate $\sigma = 1/4q$. The algorithm N we describe below has white box access to the local algorithm M'. We next explain how it extracts information from it.

Upon execution, M' chooses a decision tree uniformly at random according to the outcome of its coin flips; this uniform distribution is denoted $\tilde{\mu} = \tilde{\mu}^{M'}$, whose support size is $|\tilde{\mu}|$. For every decision tree and every one of its branches, define a *description tuple* (S, a_S, b, s), where s is the random string that will cause the use of this tree, S is the set of all the queries in this branch, a_S is the assignment to these queries that will result in M' using this specific branch and b is the value M' returns when this occurs (as per Definition 3.2).

We assume that for every description-tuple (S, a_S, b, s) the size of S is exactly q. This can be assumed without loss of generality since it is possible to convert M' into an algorithm such that every decision tree and every one of its branches makes q distinct queries: if the same query appears more than once on a branch of a tree, all but the first appearance can be removed by choosing the continuation that follows the (already known) value that leads to the algorithm using this branch. In addition, a tree can be expanded by adding queries, so that every branch has exactly q distinct queries. Both of these changes do not change any of the parameters of the algorithm beyond ensuring that it will query exactly q coordinates.

The algorithm N we describe next only makes use of description tuples (S, a_S, b, s) such that $b = 1$. To this end we set

$$\mathcal{T} = \{(S, a_S, b, s) : (S, a_S, b, s) \text{ is a description tuple such that } b = 1\}.$$

Algorithm N also requires the multi-collection $\mathcal{S}$ defined as follows:

$$\mathcal{S} = \{S : (S, a_S, b, s) \in \mathcal{T}\}.$$

Specifically, it applies Lemma 5.2 to get a daisy partition of $\mathcal{S}$. When the algorithm extracts $\mathcal{T}$ and $\mathcal{S}$ from M' and computes a daisy partition for $\mathcal{S}$, it preserves the information that allows it to associate the set of a tuple of (S, a_S, b, s) to the unique daisy S is contained in.

The construction proceeds in two stages: *preprocessing* and *execution*. Recall that, for any input $x \in \Sigma^n$ and assignment κ to a subset $S \subseteq [n]$, we denote by x_κ the word that assigns the same values as κ on S and the same values as x on $[n] \setminus S$.

Preprocessing. N has access to M', with which it computes $\mathcal{T}$ and $\mathcal{S}$. Applying Lemma 5.2 to $\mathcal{S}$, the algorithm obtains the *daisy partition*

$$\mathcal{D} = \{\mathcal{D}_j : 0 \le j \le q\},$$

so that each tuple in $\mathcal{T}$ is associated with $\mathcal{D}_j$ for exactly one $j \in \{0, \ldots, q\}$. Set

$$p \coloneqq \gamma \cdot n^{-1/2q^2},$$

the *sampling probability*, where $\gamma = 24|\Sigma|^q \ln|\Sigma|$; and, for every $j \in [q]$, set

$$\tau_j \coloneqq \frac{|\tilde{\mu}|}{2q} \cdot p^j,$$

the *thresholds*, which will be used in the execution stage.

Execution. When N receives query access to a string $x \in \Sigma^n$, it performs the following sequence of steps.

1. *Sampling:* Select each element in $[n]$ independently with probability p. Denote by Q the set of all coordinates thus obtained. If $|Q| \geq 2pn$, then N outputs arbitrarily. Otherwise, N queries all the coordinates in Q.
2. *Enumeration:* For every $j \in [q]$ and kernel assignment κ to K_j,[23] perform the following steps. Set a counting variable v to 0 before each iteration.
   (a) *Local view generation and vote counting:* For every tuple $(S, a_S, 1, s) \in \mathcal{T}$ such that $S \in \mathcal{D}_j$, increment v if $S \subset Q \cup K_j$ and a_S assigns on S the same values as x_κ does. In the case $j = 1$, if $12 \ln|\Sigma|/(\rho \cdot \sigma)$ sets have the same point outside K_1, disregard them in the count.[24]
   (b) *Threshold check:* If $v \geq \tau_j$, output 1.
3. If the condition $v \geq \tau_j$ was never satisfied, then output 0.

We proceed to analyse this construction.

6.2 Analysis

We remind that the explicit input z is assumed to be fixed and is omitted from the notation. For the analysis we are interested in the behaviour of the algorithm M' on a fixed input x. For this purpose, we use the distribution μ_x from Definition 3.2.

For $x \in \Sigma^n$ we define μ_x to be the uniform distribution over the multi-collection of sets

$$\left\{S : (S, a_S, b, s) \text{ is a description tuple such that } a_S = x_{|S}\right\}, \tag{6.1}$$

where a description tuple is as appears in Section 6.1. We note that this implies that $\mathrm{supp}(\mu_x)$ has exactly one set for each decision tree M' may use, since when both the randomness and the input are fixed exactly one branch of the decision tree is used by M'. Therefore,

$$|\mu_x| = |\tilde{\mu}|.$$

We now list the relevant parameters in the analysis with reference to where they are obtained. By Lemma 5.10,

$$\sigma = \frac{1}{4q}, \tag{6.2}$$

[23] Note that the algorithm *does not* iterate over the case $j = 0$. We will show in Section 6.2 that this has a negligible effect.

[24] This is required for technical purposes when dealing with K_1.

and, for every $x \in \Sigma^n$,

$$|\mu_x| = |\tilde{\mu}| = \frac{6n \ln |\Sigma|}{\sigma} \,. \tag{6.3}$$

The construction of N in the previous section sets the parameters

$$\gamma = 24|\Sigma|^q \ln |\Sigma| \,, \tag{6.4}$$

$$p = \gamma \cdot n^{-1/2q^2} \,, \tag{6.5}$$

and, for all $j \in [q]$,

$$\tau_j = \frac{|\tilde{\mu}|}{2q} \cdot p^j = \frac{|\mu_x|}{2q} \cdot p^j \,. \tag{6.6}$$

(The second equality holds for all $x \in \Sigma^n$ by Eq. (6.3).) Finally, the size of the collection of tuples $\mathcal{T}$, which by the construction in Section 5.4 is the same as that of $\mathcal{S}$, is bounded by the total number of branches over all decision trees in $\mathrm{supp}(\tilde{\mu})$. Thus,

$$|\mathcal{S}| = |\mathcal{T}| \leq |\Sigma|^q \cdot |\tilde{\mu}| = |\Sigma|^q \cdot |\mu_x| \,, \tag{6.7}$$

for every $x \in \Sigma^n$.

For our result we need an upper bound on the sizes of the kernels that algorithm N enumerates over, which we show next.

Claim 6.2. *Let $\{K_i : 0 \leq i \leq q\}$ be the kernels of the daisy partition $\{\mathcal{D}_i\}$ of $\mathcal{S}$ used by the algorithm N. For every $i \in \{0, 1, \ldots, q\}$, the kernel K_i is such that $|K_i| \leq \gamma \cdot q^2 \cdot n^{1-\max\{1,i\}/q}$ and, for n sufficiently large, $|K_i| < \rho n/2$.*

Proof. By Lemma 5.2, for every $i \in \{0, 1, \ldots, q\}$,

$$\begin{aligned}
|K_i| &\leq q|\mathcal{S}| n^{-\max\{1,i\}/q} \\
&\leq q|\Sigma|^q |\mu_x| \cdot n^{-\max\{1,i\}/q} && \text{(by Eq. (6.7), } |\mathcal{S}| \leq |\Sigma|^q \cdot |\mu_x|) \\
&\leq q|\Sigma|^q \cdot \frac{6n \ln |\Sigma|}{\sigma} \cdot n^{-\max\{1,i\}/q} && \left(\text{by Eq. (6.3), } |\mu_x| \leq \frac{6n \ln |\Sigma|}{\sigma}\right) \\
&= 24|\Sigma|^q \cdot \ln |\Sigma| \cdot q^2 \cdot n^{-\max\{1,i\}/q} && \left(\text{by Eq. (6.2), } \sigma = \frac{1}{4q}\right) \\
&= \gamma \cdot q^2 \cdot n^{1-\max\{1,i\}/q} && \text{(by Eq. (6.4), } \gamma = 24|\Sigma|^q \ln |\Sigma|)
\end{aligned}$$

It remains to prove the second part of the claim. By the calculation above, since ρ is constant and $|\Sigma|^q \ln |\Sigma| \cdot q^2 = o(n^{-1/q})$ (recall that $|\Sigma| \leq n^{1/q^4}$ and q is sub-logarithmic), for sufficiently large n,

$$|K_0| \leq \gamma \cdot q^2 \cdot n^{1-1/q} = \left(24|\Sigma|^q \ln |\Sigma| \cdot q^2 \cdot n^{-1/q}\right) n \leq \rho n/2. \tag{6.8}$$

By Lemma 5.2, $K_q \subseteq K_{q-1}, \ldots, K_1 = K_0$, and hence the claim follows. □

Next, we provide a number of definitions emanating from algorithm N. We define, for every $x \in \Sigma^n$, the multi-collection

$$\mathcal{O}^x \coloneqq \left\{S : (S, a_S, 1, s) \in \mathcal{T} \text{ and } x_{|S} = a_S\right\},$$

where $\mathcal{T}$ is defined as in Section 6.1. Note that the definition of this collection depends only on the algorithm M and not on the function f it computes. Hence, it is well-defined for every x, and in particular for points that are ρ-close to a ρ-robust point of the domain (where f may not be defined). We note that, since μ_x is defined over the collection in Eq. (6.1) we know that

$$\mathcal{O}_x \subseteq \mathrm{supp}(\mu_x)\,. \tag{6.9}$$

Since the "capping parameter" $12\ln|\Sigma|/(\rho\cdot\sigma)$ is used numerous times, we set

$$\alpha = \frac{12\ln|\Sigma|}{\rho\cdot\sigma}\,. \tag{6.10}$$

We refer to the act of incrementing v as counting a vote. For each $j\in[q]$, we define the *vote counting function* $v_j\colon \Sigma^n\to\mathbb{N}$ to be a random variable (determined by Q) as follows. If $j>1$,

$$v_j(x) \coloneqq \left|\{S\in\mathcal{O}^x\cap\mathcal{D}_j : S\subseteq Q\cup K_j\}\right|,$$

and $v_1(x)$ is defined likewise, with the exception that, when more than α sets intersect in a point outside K_1, they are discarded.

Claim 6.3. *Let $x\in\Sigma^n$, $j\in[q]$ and κ an assignment to K_j. Then $v_j(x_\kappa)$ is equal (as a function of Q) to the maximum value of the counter v computed by N on input x with kernel K_j and the kernel assignment κ to K_j.*

Proof. Fix $x\in\Sigma^n$. Recall that when algorithm N computes v for a $j\in\{2,3,\ldots,q\}$ and a kernel assignment κ to K_j in Step 2a, it only increases v if it encounters a tuple $(S,a_S,1,s)$ where $S\in\mathcal{D}_j$, $S\subset Q\cup K_j$ and a_S assigns on S the same values as x_κ does. Thus, by the definition of $\mathcal{O}_x$, the algorithm N counts exactly all the tuples $(S,a_S,1,s)$ such that $S\in\mathcal{O}^x$ and $S\subset Q\cup K_j$. These are precisely the sets that comprise the collection whose cardinality is $v_j(x_\kappa)$. Note that the same holds when $j=1$ due to the additional condition in Step 2a and the corresponding restriction in the definition of $v_1(x_\kappa)$. □

We now proceed to the main claims. The algorithm N only counts votes for output 1, i.e. tuples with 1 as their third value, and hence it suffices to prove that: (1) *when $f(x)=1$ and the kernel assignment is $\kappa=x_{|K_j}$* (the value of x on the indices in K_j) *for some daisy $\mathcal{D}_j$*, the number of votes is high enough to cross the threshold τ_j; and that (2) *when $f(x)=0$, then every kernel assignment κ* is such that the number of votes is smaller than the threshold. These conditions are shown to hold with high probability in Sections 6.3 and 6.4, respectively, and we show how the theorem follows from them in Section 6.5.

6.3 Correctness on non-robust inputs

Claim 6.4. *Let Q be the coordinates sampled by N and fix $x\in\Sigma^n$ such that $f(x)=1$. There exists $j\in[q]$ such that, with the kernel assignment $\kappa=x_{|K_j}$, the vote counting function satisfies $v_j(x_\kappa)=v_j(x)\geq\tau_j$ with probability at least $9/10$.*

Proof. For ease of notation, let us fix x as in the statement and denote $\mathcal{O}\coloneqq\mathcal{O}^x=\mathcal{O}^{x_\kappa}$. When $j>1$, define the subcollection of $\mathcal{O}$ in $\mathcal{D}_j$ by $\mathcal{O}_j\coloneqq\mathcal{O}^x\cap\mathcal{D}_j$; when $j=1$, define $\mathcal{O}_1\coloneqq(\mathcal{O}^x\cap\mathcal{D}_j)\setminus\mathcal{C}$, where $\mathcal{C}\subseteq\mathcal{O}^x\cap\mathcal{D}_1$ contains every $S\in\mathcal{O}^x\cap\mathcal{D}_1$ for which there exist at least $\alpha-1$ other sets in

$S' \in \mathcal{O}^x \cap \mathcal{D}_1$ that have the same petal as S, i.e., such that $S \setminus K_1 = S' \setminus K_1$. We also take n to be sufficiently large when necessary for an inequality to hold.

For the claim to hold we require the existence of $j \in [q]$ such that $\mathcal{O}_j$ is a sufficiently large portion of $\mathcal{O}$. Since $\bigcup_{0<j\le q} \mathcal{O}_j = \mathcal{O} \setminus (\mathcal{O}_0 \cup \mathcal{C})$, in order to achieve this goal, we only need to bound the sizes of both $\mathcal{O}_0$ and $\mathcal{C}$. As a first step, we bound $\mu_x(\mathcal{O}_0)$ and $\mu_x(\mathcal{C})$, which give us a lower bound on $\mu_x\left(\bigcup_{0<j\le q} \mathcal{O}_j\right)$, which we then use in order to lower bound $\left|\bigcup_{0<j\le q} \mathcal{O}_j\right|$.

We start with $\mu_x(\mathcal{O}_0)$. All the sets in $\mathcal{D}_0$ are subsets of K_0, and $|K_0| < \rho n/2$ by Claim 6.2. This implies that the cardinality of $\cup\mathcal{O}_0$ (the union of all sets in $\mathcal{O}_0$) is strictly less than ρn, and consequently, by the volume lemma (Lemma 5.6, which applies because $f(x) = 1$), we have $\mu_x(\mathcal{O}_0) \le 2\sigma$.

We now proceed to bound $\mu_x(\mathcal{C})$. As $\mathcal{C} \subseteq \mathcal{D}_1$, every set in $\mathcal{C}$ has exactly one element in $[n] \setminus K_1$ and repeats at least α times, the cardinality of $\cup\mathcal{C}$ is at most $|K_1| + \frac{|\mathcal{C}|}{\alpha}$. By Claim 6.2, $|K_1| < \rho n/2$, and it follows that

$$\begin{aligned}
|K_1| + \frac{|\mathcal{C}|}{\alpha} &< \frac{\rho n}{2} + \frac{|\mathcal{O}|}{\alpha} \\
&\le \frac{\rho n}{2} + \frac{|\mu_x|}{\alpha} && \text{(by Eq. (6.9), } \mathcal{O} = \mathcal{O}_x \subseteq \operatorname{supp}(\mu_x)) \\
&\le \frac{\rho n}{2} + |\mu_x| \cdot \frac{\rho \cdot \sigma}{12 \ln |\Sigma|} && \left(\text{by Eq. (6.10), } \alpha^{-1} = \frac{\rho \cdot \sigma}{12 \ln |\Sigma|}\right) \\
&= \frac{\rho n}{2} + \frac{6n \ln |\Sigma|}{\sigma} \cdot \frac{\rho \cdot \sigma}{12 \ln |\Sigma|} && \left(\text{by Eq. (6.3), } |\mu_x| = \frac{6n \ln |\Sigma|}{\sigma}\right) \\
&\le \rho n \,.
\end{aligned}$$

Consequently, by Lemma 5.6, $\mu_x(\mathcal{C}) \le 2\sigma$.

By the definition of error rate, $\mu_x(\mathcal{O}) \ge 1 - \sigma$. Since $\{\mathcal{O}_j : 0 \le j \le q\}$ is a partition of $\mathcal{O}$ (because $\{\mathcal{D}_j\}$ is a partition),

$$\mu_x\left(\bigcup_{0<j\le q} \mathcal{O}_j\right) = \mu_x(\mathcal{O}) - \mu_x(\mathcal{O}_0) - \mu_x(\mathcal{C}) \ge 1 - 5\sigma \,.$$

As μ_x is uniform, each element of the multi-collection $\mathcal{O}$ has weight exactly $1/|\mu_x|$. Therefore,

$$\sum_{j=1}^{q} |\mathcal{O}_j| = |\mu_x| \cdot \mu_x(\cup_{j\in[q]}\mathcal{O}_j) \ge |\mu_x|(1 - 5\sigma) \ge |\mu_x|/2 \,,$$

where the last inequality follows from the assumption that $5\sigma \le 1/2$ (which follows, e.g., from $q \ge 3$, which Lemma 5.10 ensures). Let j be such that

$$|\mathcal{O}_j| \ge \frac{|\mu_x|}{2q} \,; \tag{6.11}$$

by averaging, such a j indeed exists. Our goal now is to show that with probability at least $9/10$, there are at least τ_j sets $S \in \mathcal{O}_j$ whose petal is in Q, i.e., such that $S \setminus K_j \subseteq Q$.

Instead of proving this directly on $\mathcal{O}_j$, we do so on collections that form a partition of $\mathcal{O}_j$ and have a useful structure. The sets in $\mathcal{O}_j$ are also in $\mathcal{D}_j$, so that $\mathcal{O}_j$ is also a daisy with kernel K_j.

By Claim 5.3, for every set $S \in \mathcal{O}_j$, there exist at most $2n^{\max\{1,j-1\}/q} - 1$ sets $S' \in \mathcal{O}_j \setminus \{S\}$ whose petals have a non-empty intersection with the petal of S, i.e, such that $(S \cap S') \setminus K_j \neq \varnothing$. This enables us to apply Lemma 5.5 to $\mathcal{O}_j$, partitioning it into $\{\mathcal{S}_i : i \in [t]\}$ simple daisies of equal sizes, where

$$t \leq 2n^{\max\{1,j-1\}/q} . \tag{6.12}$$

Thus, for every $i \in [t]$,

$$|\mathcal{S}_i| = \frac{|\mathcal{O}_j|}{t} . \tag{6.13}$$

Let $\mathcal{O}'_j$ be the multi-collection of all sets $S \in \mathcal{O}_j$ such that $S \setminus K_j \subseteq Q$. In the same manner, for every $i \in [t]$, let $\mathcal{S}'_i$ be the multi-collection of all sets $S \in \mathcal{S}_i$ such that $S \setminus K_j \subseteq Q$.

By construction, the collections $\{\mathcal{S}'_i\}$ are pairwise disjoint. Also, by the definition of v_j, we have $v_j(x) = \left|\mathcal{O}'_j\right| = \sum_{i=1}^{t} |\mathcal{S}'_i|$. Therefore, the event that $v_j(x) \leq \tau_j$ can only occur if there exists $i \in [t]$ such that $|\mathcal{S}'_i| \leq \tau_j/t$.

Consequently, we obtain

$$\begin{aligned}
\Pr\left[v_j(x) \leq \tau_j\right] &\leq \Pr\left[\left|\mathcal{S}'_i\right| \leq \frac{\tau_j}{t} \text{ for some } i \in [t]\right] \\
&\leq \sum_{i=1}^{t} \Pr\left[\left|\mathcal{S}'_i\right| \leq \frac{\tau_j}{t}\right] && \text{(union bound)} \\
&\leq t \Pr\left[\left|\mathcal{S}'_1\right| \leq \frac{\tau_j}{t}\right] . && \text{(all } \mathcal{S}_i \text{ have equal size)}
\end{aligned}$$

We show afterwards that the probability of the event $|\mathcal{S}'_1| \leq \frac{\tau_j}{t}$ is strictly less than $1/10t$, which by the inequality above implies the claim.

We later use the Chernoff bound with $\mathcal{S}_1$, and hence we start by bounding $\mathbb{E}[|\mathcal{S}'_1|]$ from below. Recall that the petal of every set $S \in \mathcal{S}_1 \subseteq \mathcal{D}_j$ has size j (i.e., $|S \setminus K_j| = j$), and therefore is in $\mathcal{S}'_1$ with probability exactly p^j. So

$$\begin{aligned}
\mathbb{E}[\left|\mathcal{S}'_1\right|] = |\mathcal{S}_1| p^j &= \frac{|\mathcal{O}_j| p^j}{t} && \text{(by Eq. (6.13), } |\mathcal{S}_1| = |\mathcal{O}_j|/t) \\
&\geq \frac{|\mu_x| p^j}{2tq} && \left(\text{by Eq. (6.11), } |\mathcal{O}_j| \geq \frac{|\mu_x|}{2q}\right) \quad (6.14) \\
&= \frac{\tau_j}{t}. && \left(\text{by Eq. (6.6), } \tau_j = \frac{|\mu_x|}{2q} \cdot p^j\right)
\end{aligned}$$

Thus,

$$\Pr\left[\left|\mathcal{S}'_1\right| \leq \frac{1}{2}\mathbb{E}[\left|\mathcal{S}'_1\right|]\right] \geq \Pr\left[\left|\mathcal{S}'_1\right| \leq \frac{\tau_j}{t}\right] .$$

Next we show that the probability of the event $|\mathcal{S}'_1| \leq \frac{1}{2}\mathbb{E}[|\mathcal{S}'_1|$ is at most $1/10t$, which concludes the proof. Since $\mathcal{S}_1$ is a simple daisy, the petals of sets in $\mathcal{S}_1$ are pairwise disjoint and hence the events $S \setminus K_j \subset Q$, for every $S \in \mathcal{S}_1$, are all independent. This enables us to use the Chernoff bound to

get that

$$
\begin{aligned}
&\Pr\left[|\mathcal{S}_1'| \le \frac{1}{2}\mathbb{E}[|\mathcal{S}_1'|]\right] \\
&\quad\le \exp\left(-\frac{\mathbb{E}[|\mathcal{S}_1'|]}{8}\right) && \text{(Chernoff bound)} \\
&\quad\le \exp\left(-\frac{|\mu_x|p^j}{16tq}\right) && \text{(by Eq. (6.14))} \\
&\quad\le \exp\left(-\frac{6n\ln|\Sigma|}{\sigma}\cdot\frac{p^j n}{16tq}\right) && \left(\text{by Eq. (6.3)}, |\mu_x| = \frac{6n\ln|\Sigma|}{\sigma}\right) \\
&\quad\le \exp\left(-4q\cdot\frac{3\ln|\Sigma|p^j n}{8tq}\right) && \left(\text{by Eq. (6.2)}, \sigma^{-1} = 4q\right) \\
&\quad\le \exp\left(-\frac{3\ln|\Sigma|p^j}{4}\cdot n^{1-\frac{\max\{1,j-1\}}{q}}\right) && \left(\text{by Eq. (6.12)}, t \le 2n^{\frac{\max\{1,j-1\}}{q}}\right) \\
&\quad\le \frac{1}{10t}\exp\left(-\gamma^j\cdot\frac{3\ln|\Sigma|}{4}\cdot n^{1-\frac{\max\{1,j-1\}}{q}-\frac{j}{2q^2}} + \ln(20t)\right) && \left(p = \gamma\cdot n^{-1/2q^2}\right) \\
&\quad\le \frac{1}{10t}\exp\left(-\gamma\cdot\frac{3\ln|\Sigma|}{4}\cdot n^{\frac{1}{q}-\frac{1}{2q}} + \ln(10t)\right) && (1 \le j \le q) \\
&\quad\le \frac{1}{10t}\,,
\end{aligned}
$$

where the last inequality follows for n sufficiently large because $\ln t \le \frac{\max\{1,j-1\}}{q}\ln n + 1 = o(n^{1/2q})$ and $\gamma\cdot\ln|\Sigma| = \Omega(1)$. □

Note that, although a success probability of 9/10 suffices to ensure correctness of a single run of N, Claim 6.4 yields a much stronger result: the failure probability is *exponentially small*. This is because Claim 6.4 does not enumerate over kernel assignments. Moreover, the analysis for the case $j = 1$ can be improved significantly (as will be necessary in Claim 6.5), but this does not yield in an overall improvement in our results.

6.4 Correctness on robust inputs

In the following claim we note that $|K_1|/n$-robustness suffices for the analysis, since it ensures all kernel assignments κ lead x_κ to also output $f(x) = 0$.

Claim 6.5. *Suppose the input $x \in \Sigma^n$ is $|K_1|/n$-robust for M' and $f(x) = 0$. Then, for every $j \in [q]$ and every assignment κ to the kernel K_j, the vote count satisfies $v_j(x_\kappa) < \tau_j$ with probability at least $1 - |\Sigma|^{|K_j|}/(10q)$.*

Proof. For ease of notation, fix $j \in [q]$, an assignment κ to K_j and x as in the statement, and let $\mathcal{O} := \mathcal{O}^{x_\kappa}$. If $j > 1$, define the subcollection of $\mathcal{O}$ in $\mathcal{D}_j$ by $\mathcal{O}_j := \mathcal{O} \cap \mathcal{D}_j$; if $j = 1$, define $\mathcal{O}_1 := (\mathcal{O} \cap \mathcal{D}_1) \setminus \mathcal{C}$, where $\mathcal{C} \subseteq \mathcal{O} \cap \mathcal{D}_1$ contains every $S \in \mathcal{O} \cap \mathcal{D}_1$ for which there exist at least $\alpha - 1$ other sets $S' \in \mathcal{O} \cap \mathcal{D}_1$ that have the same petal as S, i.e., such that $S \setminus K_1 = S' \setminus K_1$. We also take n to be sufficiently large when necessary for an inequality to hold.

Note that x_κ may not be in the domain of f, but the robustness of x allows us to bound the size of $\mathcal{O} = \mathcal{O}^{x_\kappa}$ regardless. Moreover, since $f(x) = 0$, we know that $\mu_x(\mathcal{O}) \leq \sigma$. As μ_x is uniform, each element of the multi-collection has $\mathcal{O}$ weight exactly $1/|\mu_x|$. Therefore, for every $i \in [q]$,

$$|\mathcal{O}_i| \leq \sigma |\mu_x| \,. \tag{6.15}$$

Our goal now is, for every $j \in [q]$, to upper bound the probability that there are at least τ_j sets $S \in \mathcal{O}_j$ whose petal is in Q, i.e., such that $S \setminus K_j \subseteq Q$.

For every $j \in [q]$, let β_j be such that for every set $S \in \mathcal{O}_j$ there exist at most $\beta_j - 1$ other distinct sets $S' \in \mathcal{O}_j$ whose petal intersects the petal of S, i.e., $(S \setminus K_1) \cap (S' \setminus K_1) \neq \varnothing$.

For the time being let us fix $j \in [q]$. By applying Lemma 5.5, we partition $\mathcal{O}_j$ into $\{\mathcal{S}_i : i \in [\beta_j]\}$, such that for every $i \in [q]$,

$$|\mathcal{S}_i| = \frac{|\mathcal{O}_i|}{\beta_j} \leq \frac{\sigma |\mu_x|}{\beta_j} \,, \tag{6.16}$$

where the inequality follows from Eq. (6.15), and each $\mathcal{S}_i$ is a simple daisy of size $|\mathcal{O}_1|/\beta_j$.

Let $\mathcal{O}'_j$ be the multi-collection of all sets $S \in \mathcal{O}_j$ such that $S \setminus K_j \subseteq Q$. In the same manner, for every $i \in [\beta_j]$, let $\mathcal{S}'_i$ be the multi-collection of all sets $S \in \mathcal{S}_i$ such that $S \setminus K_j \subseteq Q$. By the definition of v_j and the fact that $\{\mathcal{S}_i\}$ is a partition

$$v_j(x_\kappa) = \left|\mathcal{O}'_j\right| = \sum_{i=1}^{\beta_j} \left|\mathcal{S}'_i\right|.$$

Since the event $v_1(x_\kappa) \geq \tau_j$ can only occur if $|\mathcal{S}'_i| \geq \frac{\tau_j}{\beta_j}$ for some $i \in [\beta_j]$, we obtain

$$\begin{aligned}
\Pr\left[v_j(x) \geq \tau_j\right] &\leq \Pr\left[\left|\mathcal{S}'_i\right| \geq \frac{\tau_j}{\beta_j} \text{ for some } i \in [\beta_j]\right] \\
&\leq \sum_{i=1}^{\beta_j} \Pr\left[\left|\mathcal{S}'_i\right| \geq \frac{\tau_j}{\beta_j}\right] && \text{(union bound)} \\
&\leq \beta_j \cdot \Pr\left[\left|\mathcal{S}'_1\right| \geq \frac{\tau_j}{\beta_j}\right] \,. && \text{(all } \mathcal{S}_i \text{ have equal size)}
\end{aligned}$$

Now our goal is to show that the event that $|\mathcal{S}'_1| \geq \frac{\tau_j}{\beta_j}$ happens with probability at most $\frac{|\Sigma|^{-|K_1|}}{10q\beta_j}$. Note that this is sufficient for proving the claim because plugging this into the previous equation gives $\Pr\left[v_j(x) \geq \tau_j\right] \leq |\Sigma|^{-|K_j|}/(10q)$.

Since the sets in $\mathcal{S}_1$ are pairwise disjoint, we can and do use the Chernoff bound. In order to do so we first bound the value of $\mathbb{E}[|\mathcal{S}'_1|]$ from above. Recall that the petal of every set $S \in \mathcal{S}_j \subseteq \mathcal{D}_j$ has size j (i.e., $|S \setminus K_j| = j$), and therefore S is in $\mathcal{S}'_j$ with probability exactly p^j. So,

$$\begin{aligned}
\mathbb{E}[\left|\mathcal{S}'_1\right|] &= |\mathcal{S}_1| \cdot p^j \\
&\leq \frac{\sigma \cdot |\mu_x| \cdot p^j}{\beta_j} && \text{(by Eq. (6.16))} \\
&= \frac{\tau_j}{2\beta_j} \,. && \left(\text{by Eq. (6.2) and Eq. (6.6), } \sigma = \frac{1}{4q} \text{ and } \tau_j = \frac{|\mu_x|}{2q} \cdot p^j\right)
\end{aligned}$$

We now use the Chernoff bound, stopping at a partial result and providing separate analyses for the cases $j = 1$ and $j > 1$.

$$\begin{aligned}
\Pr\left[|\mathcal{S}'_1| \geq \frac{\tau_j}{\beta_j}\right] &= \Pr\left[|\mathcal{S}'_1| \geq \frac{\tau_j}{\beta_j \mathbb{E}[|\mathcal{S}'_1|]}\mathbb{E}[|\mathcal{S}'_1|]\right] \\
&\leq \exp\left(-\left(\frac{\tau_j}{\beta_j \mathbb{E}[|\mathcal{S}'_1|]} - 1\right)^2 \cdot \frac{\mathbb{E}[|\mathcal{S}'_1|]}{3}\right) && \text{(Chernoff bound)} \\
&\leq \exp\left(-\frac{\tau_j}{3\beta_j}\right) && \text{(explained aferwards)} \\
&= \exp\left(-\frac{|\mu_x| \cdot p^j}{6q\beta_j}\right) && \left(\text{by Eq. (6.6), } \tau_j = \frac{|\mu_x|}{2q} \cdot p^j\right) \\
&= \exp\left(-\frac{\ln|\Sigma| n p^j}{q\beta_j \cdot \sigma}\right), && \left(\text{by Eq. (6.3), } |\mu_x| = \frac{6n\ln|\Sigma|}{\sigma}\right)
\end{aligned}$$

where the second inequality follows from $\left(\frac{\tau_j}{\beta_j \mathbb{E}[|\mathcal{S}'_1|]} - 1\right)^2 \cdot \frac{\mathbb{E}[|\mathcal{S}'_1|]}{3}$ being minimal when $\mathbb{E}[|\mathcal{S}'_1|]$ is at its upper bound of $\frac{\tau_j}{2\beta_j}$. We next proceed to the first of the two cases.

Take $j = 1$. In this case, by the construction of the daisy partition (Lemma 5.2), every set $S \in \mathcal{O}_1$ has a petal $S \setminus K_1$ of cardinality exactly 1. By the definition of $\mathcal{O}_1$, each set $S \in \mathcal{O}_1$ has at most $\alpha - 1$ other sets $S' \in \mathcal{O}_1$ whose petal intersects the petal of S, i.e., $(S \setminus K_1) \cap (S' \setminus K_1) \neq \varnothing$ (and thus $S \setminus K_1 = S' \setminus K_1$, since both petals have size 1). Therefore, at most $\beta_1 - 1 = \alpha - 1$ distinct sets of $\mathcal{O}_1$ intersect each $S \in \mathcal{O}_1$, which follows from Eq. (6.10). Now,

$$\begin{aligned}
&\exp\left(-\frac{\ln|\Sigma| n p}{q\alpha \cdot \sigma}\right) \\
&= \exp\left(-\frac{n \cdot p \cdot \rho}{12q}\right) && \left(\text{by Eq. (6.10), } \alpha = \frac{12\ln|\Sigma|}{\rho \cdot \sigma}\right) \\
&= \exp\left(-\gamma \cdot \frac{\rho}{12q} \cdot n^{1-1/2q^2}\right) && \left(p = \gamma \cdot n^{-1/2q^2}\right) \\
&= \frac{1}{10q}\exp\left(-\gamma \cdot n^{1-1/q} \cdot \frac{\rho \cdot n^{\frac{1}{q}-\frac{1}{2q^2}}}{12q} + \ln(10q)\right) \\
&\leq \frac{1}{10q}\exp\left(-\ln|\Sigma| \cdot \gamma \cdot q^2 \cdot n^{1-1/q}\right) && \text{(large enough } n\text{)} \\
&\leq \frac{|\Sigma|^{-|K_1|}}{10q},
\end{aligned}$$

where the last inequality follows because $|K_1| \leq \gamma \cdot q^2 \cdot n^{1-1/q}$ by Claim 6.2 (and $\ln|\Sigma| \leq \log n$).

Now, take $j > 1$. By Claim 5.3, $\beta_j = 2h(j-1) = 2n^{(j-1)/q}$, which implies the first equality in

the following.

$$\begin{aligned}
\exp\left(-\frac{\ln|\Sigma|\cdot np^j}{q\beta_j\cdot\sigma}\right) &= \exp\left(-\frac{\ln|\Sigma|\cdot np^j}{2q\cdot\sigma}\cdot n^{-\frac{j-1}{q}}\right)\\
&= \exp\left(-2\ln|\Sigma|\cdot p^j\cdot n^{1-\frac{j-1}{q}}\right) && \left(\text{by Eq. (6.2), } \sigma=\frac{1}{4q}\right)\\
&= \exp\left(-2\ln|\Sigma|\cdot \gamma^j\cdot n^{1-\frac{j-1}{q}-\frac{j}{2q^2}}\right) && \left(p=\gamma\cdot n^{-1/2q^2}\right)\\
&= \exp\left(-2\ln|\Sigma|\cdot \gamma^j\cdot n^{1-\frac{j}{q}+\frac{2q-j}{2q^2}}\right)\\
&\le \frac{1}{10q}\exp\left(-\ln|\Sigma|\cdot\gamma\cdot n^{1-\frac{j}{q}}\cdot 2n^{\frac{1}{2q}}+\ln(10q)\right) && (1<j\le q)\\
&\le \frac{1}{10q}\exp\left(-\ln|\Sigma|\cdot\gamma\cdot q^2\cdot n^{1-j/q}\right) && (\text{large enough } n)\\
&\le \frac{|\Sigma|^{-|K_j|}}{10q},
\end{aligned}$$

where the last inequality follows because $|K_j|\le\gamma\cdot q^2\cdot n^{1-j/q}$ by Claim 6.2. □

6.5 Concluding the proof

We conclude the proof Theorem 6.1 by applying the two previous claims. Recall that we transformed a ρ-robust local algorithm M for a function f, with query complexity ℓ, into a ρ-robust local algorithm M' with query complexity $q=O(\ell\log\ell)$ and suitable error rate. Then we transformed M' into a sample-based algorithm N with sample complexity $n^{1-1/O(q^2)}=n^{1-1/O(\ell^2\log^2\ell)}$, an upper bound guaranteed by the sampling step (Step 1) in the construction of N. It remains to show correctness of the algorithm on every input in the domain of f.

We first consider errors that may arise in the sampling step. By the Chernoff bound, it chooses more than $2pn=2n^{1-j/(2q^2)}$ points to query and thus outputs arbitrarily with probability at most $1/10$. Otherwise, it proceeds to the next steps.

In the next part of the proof we analyse $v_j(x)$ instead of analyzing v (of Step 2a) in algorithm N; this is sufficient, since by Claim 6.3, they are distributed identically over Q.

Suppose the input $x\in\Sigma^n$ is such that $f(x)=0$. Since x is ρ-robust, it is in particular $|K_1|/n$-robust (because $|K_1|=o(n)$). Then Claim 6.5 ensures that, for every $j\in[q]$ and kernel assignment κ to K_j, the vote counter satisfies $v_j(x_\kappa)\ge\tau_j$ with probability at most $|\Sigma|^{-|K_j|}/(10q)$. A union bound over all $j\in[q]$ and $|\Sigma|^{|K_j|}$ assignments to the kernel K_j ensures the probability this happens, causing N to output 1 in the threshold check step (Step 2b), is at most $1/10$; otherwise, N will enumerate over every assignment and then (correctly) output 0 in Step 3.

Now suppose $x\in\Sigma^n$ is such that $f(x)=1$. Then Claim 6.4 ensures that, for some $j\in[q]$, the kernel assignment $\kappa=x_{|K_j}$ will make the vote count satisfy $v_j(x)\ge\tau_j$ with probability at least $9/10$, in which case N (correctly) outputs 1 in the threshold check step (Step 2b).

Therefore, N proceeds beyond the sampling step with probability $9/10$ and outputs correctly (due to Claim 6.5 and Claim 6.4) with probability at least $9/10-1/10\ge 2/3$. This concludes the proof of Theorem 6.1.

Remark 6.6. Notice that the claims actually prove a stronger statement: the failure probability is not merely $1/3$, but *exponentially small*. For each $j \in [q]$, the error probability is

$$\exp\left(-\Omega\left(n^{1-\frac{j}{q}+\frac{2q-j}{2q^2}}\right)\right),$$

but it must withstand a union bound over $\exp\left(O\left(n^{1-j/q}\right)\right)$ events (corresponding to the assignments to the kernel K_j). The smallest slackness is in the case $j = q$, where the success probability is still $\exp\left(-\Omega\left(n^{1/2q}\right)\right)$; this implies that correctness holds for $\exp\left(c \cdot n^{1/2q}\right)$ many executions, if the constant c is sufficiently small. Therefore, *the same samples can be reused for exponentially many runs of possibly different algorithms.*

7 Applications

In this section, we derive applications from Theorem 6.1 which range over three fields of study: property testing, coding theory, and probabilistic proof systems. We first give a brief overview of the applications in the following paragraphs, then proceed to the proofs in Sections 7.1 to 7.3.

Query-to-sample tradeoffs for adaptive testers. The application to property testing is an immediate corollary of Theorem 6.1: since an $\varepsilon/2$-tester is a $(\varepsilon/2, 0)$-robust algorithm for the problem of testing with proximity parameter $\varepsilon/2$, Corollary 7.1 shows that any $\varepsilon/2$-tester making q adaptive queries can be transformed into a sample-based ε-tester with sample complexity $n^{1-1/O(q^2 \log^2 q)}$. In addition, we also show an application to multi-testing (Corollary 7.2).

Relaxed LDC lower bound. By a straightforward extension of our definition of robust local algorithms to allow for outputting a special failure symbol $\perp$, our framework captures *relaxed* LDCs (see Section 7.2). We remark that, although standard LDCs have *two-sided* robustness, the treatment of relaxed LDCs is analogous to one-sided robust algorithms.

By applying Theorem 6.1 to a relaxed local decoder *once for each bit to be decoded*, we obtain a *global* decoder that decodes uncorrupted codewords with $n^{1-1/O(q^2 \log^2 q)}$ queries; by a simple information-theoretic argument, we obtain a rate lower bound of $n = k^{1+1/O(q^2 \log^2 q)}$ for relaxed LDCs (see Corollaries 7.8 and 7.9).

Tightness of the separation between MAPs and testers. Theorem 6.1 applies to the setting of *Merlin-Arthur proofs of proximity* (MAPs) via a description of MAPs as coverings by partial testers (Claim 7.10). In Section 7.3, we show that the existence of an adaptive MAP for a property Π with query complexity q and proof length m implies the existence of a sample-based tester for Π with sample complexity $m \cdot n^{1-1/O(q^2 \log^2 q)}$ (Theorem 7.11).

This implies that there exists no property admitting a MAP with query complexity $q = O(1)$ and logarithmic proof length (in fact, much longer proof length) that requires at least $n^{1-1/\omega(q^2 \log^2 q)}$ queries for testers, showing the (near) tightness of the separation from [GR18].

Optimality of Theorem 6.1. We conclude Section 7.3 with a direct corollary of the tightness of the aforementioned separation between MAPs and testers of [GR18], we obtain that the general transformation in Theorem 6.1 is optimal, up to a quadratic gap in the dependency on the sample

complexity. This follows simply because a transformation with smaller sample complexity could have been used to improve Theorem 7.11, yielding a tester with query complexity that contradicts the lower bound (see Theorem 7.12).

7.1 Query-to-sample tradeoffs for adaptive testers

Recall that a property tester T for property $\Pi \subseteq \Sigma^n$ is an algorithm that receives explicit access to a proximity parameter $\varepsilon > 0$, query access to $x \in \Sigma^n$ and *approximately decides* membership in Π: it accepts if $x \in \Pi$ and rejects if x is ε-far from Π, with high probability.

By Claim 4.6, an ε-tester with $\varepsilon \in (0,1)$ is an ε-robust local algorithm that computes the function $f : \Pi \cup \overline{B_{2\varepsilon}(\Pi)} \to \{0,1\}$ defined as follows.

$$f(x) = \begin{cases} 1, & \text{if } x \in \Pi \\ 0, & \text{if } x \in \overline{B_{2\varepsilon}(\Pi)}. \end{cases}$$

Note, moreover, that a local algorithm that solves f is by definition a 2ε-tester, accepting elements of Π and rejecting points that are 2ε-far from it with high probability. A direct application of Theorem 6.1 thus yields the following corollary, which improves upon the main result of [FLV15], by extending it to the two-sided adaptive setting.

Corollary 7.1. *For every fixed $\varepsilon > 0, q \in \mathbb{N}$, any ε-testable property of strings in Σ^n with q queries admits a sample-based 2ε-tester with sample complexity $n^{1-1/O(q^2 \log^2 q)}$.*

This also immediately extends an application to multitesters in [FLV15]. By standard error reduction, for any $k \in \mathbb{N}$, an increase of the sample complexity by a factor of $O(\log k)$ ensures each member of a collection of k sample-based testers errs with probability $1/(3k)$. A union bound allows us to *reuse the same samples for all testers*, so that all will output correctly with probability $2/3$. Taking $k = \exp\left(n^{1/\omega(q^2 \log^2 q)}\right)$, the sample complexity becomes $n^{1-1/O(q^2 \log^2 q)} \cdot n^{1/\omega(q^2 \log^2 q)} = o(n)$, which yields the following corollary.

Corollary 7.2. *If a property $\Pi \subseteq \Sigma^n$ is the union of $k = \exp\left(n^{1/\omega(q^2 \log^2 q)}\right)$ properties $\Pi_1, \ldots, \Pi_k$, each ε-testable with q queries, then Π is 2ε-testable via a sample-based tester with sublinear sample complexity.*

A tester for the union simply runs all (sub-)testers, accepting if and only if at least one of them accepts. A proof for a generalisation of this corollary, which holds for *partial testers*, is given in the Section 7.3.

7.2 Stronger relaxed LDC lower bounds

Relaxed LDCs are codes that relax the notion of LDCs by allowing the local decoder to abort on a small fraction of the indices, yet crucially still avoid errors. This seemingly modest relaxation turns out to allow for dramatically better parameters (an exponential improvement on the rate of the best known $O(1)$-query LDCs). However, since these algorithms are much stronger, obtaining lower bounds on relaxed LDCs is significantly harder than on standard LDCs. Indeed, the first lower bound on relaxed LDCs [GL21] was only shown more than a decade after the notion was introduced;

this bound shows that to obtain query complexity q, a relaxed LDC $C\colon \{0,1\}^k \to \{0,1\}^n$ must have blocklength

$$n \geq k^{1+\frac{1}{O(2^{2q}\cdot \log^2 q)}} ,$$

In this section, we use Theorem 6.1 to obtain an improved lower bound with an exponentially better dependency on the query complexity. We begin by recalling the definition of relaxed LDCs.

Definition 7.3 (Definition 4.12, restated)**.** *A code $C\colon \{0,1\}^k \to \{0,1\}^n$ whose distance is δ_C is a q-local relaxed LDC with success rate ρ, decoding radius $\delta \in (0, \delta_C/2)$ and error rate $\sigma \in (0, 1/3]$ if there exists a randomised algorithm D, known as a* relaxed decoder*, that, on input $i \in [k]$, makes at most q queries to an oracle w and satisfies the following conditions.*

1. *Completeness: For any $i \in [k]$ and $w = C(x)$, where $x \in \{0,1\}^k$,*

$$\Pr[D^w(i) = x_i] \geq 1 - \sigma .$$

2. *Relaxed Decoding: For any $i \in [k]$ and $w \in \{0,1\}^n$ that is δ-close to a (unique) codeword $C(x)$,*

$$\Pr[D^w(i) \in \{x_i, \perp\}] \geq 1 - \sigma .$$

3. *Success Rate: There exists a constant $\rho > 0$ such that, for any $w \in \{0,1\}^n$ that is δ-close to a codeword $C(x)$, there exists a set $I_w \subseteq [k]$ of size at least ρk such that for every $i \in I_w$,*

$$\Pr[D^w(i) = x_i] \geq 2/3 .$$

Remark 7.4. The first two conditions imply the latter, as shown by [BGH+06]. Therefore, it is not necessary to show the success rate condition when verifying that an algorithm D is a relaxed local decoder.

Note that, whenever D^w outputs $\perp$, it detected that the input is *not valid*, since it is inconsistent with *any* codeword $C(x)$. We slightly generalise local algorithms (Definition 4.1) to capture this behaviour, by allowing them to output $\perp$ as well as the correct function evaluation $f(z, x)$ (except for a prescribed set of valid inputs). Formally,

Definition 7.5 (Relaxed local algorithm)**.** *Let Σ be a finite alphabet, Z a finite set and $\{\mathcal{P}_z : z \in Z\}$ a family of sets $\mathcal{P}_z \subseteq \Sigma^n$ indexed by Z. With $\mathcal{P} := \{(z, x) : z \in Z, x \in \mathcal{P}_z\}$, let $f\colon \mathcal{P} \to \{0,1\}$ be a partial function.*

A relaxed q-local algorithm M for computing f *with* valid input set $V \subseteq \Sigma^n$ *and* error rate σ *receives explicit access to $z \in Z$, query access to $x \in \mathcal{P}_z$, makes at most q queries to x and satisfies*

$$\Pr\left[M^x(z) \in \{f(z, x), \perp\}\right] \geq 1 - \sigma.$$

Moreover, if $x \in V$, then M satisfies

$$\Pr[M^x(z) = f(z, x)] \geq 1 - \sigma.$$

We shall also need to generalise the notion of robustness (Definition 4.2) accordingly.

Definition 7.6 (Robustness)**.** *Let $\rho > 0$. A local algorithm M for computing $f\colon \mathcal{P} \to \{0,1\}$ is* ρ-robust *at the point $(z,x) \in \mathcal{P}$ if $\Pr[M^w(z) \in \{f(z,x), \perp\}] \geq 1 - \sigma$ for all $w \in B_\rho(x)$. We say that M is* (ρ_0, ρ_1)-robust *if, for all $z \in Z$ and $b \in \{0,1\}$, M is ρ_b-robust at every x such that $f(z,x) = b$.*

We remark that robustness for algorithms that allow aborting allows the correct value to change to $\perp$ (but, crucially, not to the wrong value) even if only one bit is changed. This makes the argument more involved than an argument for LDCs, and indeed, our theorem for *relaxed* LDCs relies on the full machinery of Theorem 6.1.

Note that an algorithm that ignores its input and always outputs $\perp$ fits both definitions above, but has no valid inputs and clearly does not display any interesting behaviour. We also remark that the set of valid inputs captures completeness (but *not* the success rate) in the case of relaxed LDCs.

With these extensions, a relaxed local decoder D with decoding radius δ fits the definition of a (relaxed) local algorithm that receives $i \in [k]$ as explicit input, where the code C comprises the valid inputs and every $x \in C$ is δ-robust for D.

While a relaxed local algorithm is very similar in flavour to a standard local algorithm, it may not be entirely clear whether a transformation analogous to Theorem 6.1 holds in this case as well. We next show that one indeed does: with small modifications to the algorithm constructed in Section 6.1, we leverage the same analysis of Section 6.2 to prove the following variant of Theorem 6.1.

Theorem 7.7. *Suppose there exists a (ρ_0, ρ_1)-robust relaxed local algorithm M for computing the function $f\colon \mathcal{P} \to \{0,1\}$ (where $Z \times \Sigma^n$) with query complexity $\ell = O(1)$ and $\rho_0, \rho_1 = \Omega(1)$. Let $V \subseteq \Sigma^n$ be the valid inputs of M. Then, there exists a* sample-based *relaxed local algorithm N for f with sample complexity $n^{1-1/O(\ell^2 \log^2 \ell)}$ with the same set V of valid inputs.*

Proof. Throughout the proof, we assume the explicit input to be fixed and omit it from the notation. First, note that error reduction (Claim 5.8) and randomness reduction (Claim 5.9) apply in the relaxed setting: the analysis is identical on valid inputs, and holds likewise for the remainder of the domain of f (with correctness of M relaxed to be $M^x \in \{f(x), \perp\}$). Thus Lemma 5.10 enables the transformation of M into another robust algorithm M' with small error rate that uniformly samples a decision tree from a multi-collection of small size.

Recall that the construction of the sample-based algorithm in Section 6.1 uses a collection of triplets obtained from the behaviour of M' when it outputs 1. A corresponding collection can be obtained for the case where M' outputs 0. Denote by $\mathcal{T}_b$ the collection that corresponds to output $b \in \{0,1\}$, and let N_b be the sample-based algorithm that

- uses the triplets $\mathcal{T}_b$ to construct its daisy partition in the preprocessing step;
- outputs b if the counter crosses the threshold in Step 2b; and
- outputs $\perp$ in Step 3 if the threshold is never reached;

but is otherwise the same as the construction of Section 6.1.

The analysis of Section 6.2 applies to N_b^x: if $x \in V$, the analysis of Claim 6.4 is identical; while if x is robust and $f(x) = \neg b$, Claim 6.5 requires a lower bound on the probability that M' outputs b when its input is x (and enables an application of the volume lemma), which holds by the definition of error rate of a relaxed local algorithm. Therefore, the probability of each the following events is bounded by $1/10$:

(i) N_b^x outputs arbitrarily in the sampling step;
(ii) N_b^x outputs $\perp$ when $f(x) = b$; and

(iii) N_b^x outputs b when x is robust and $f(x) = \neg b$.

Finally, the relaxed sample-based algorithm N simply executes the sampling step of N_0 then the enumeration steps of N_0 and N_1 on these samples, outputs b if exactly one of N_b outputs b, and outputs $\perp$ otherwise. Then, $N^x = f(x)$ if $x \in V$ and $N^x = \perp$ if $x \notin V$, with probability $7/10 \geq 2/3$. □

By casting a relaxed decoder as a robust relaxed local algorithm and applying Theorem 7.7, we obtain the following corollary.

Corollary 7.8. *Any binary code $C\colon \{0,1\}^k \to \{0,1\}^n$ that admits a relaxed local decoder D with decoding radius δ and query complexity $q = O(1)$ also admits a sample-based relaxed local decoder D' with decoding radius $\delta/2$ and sample complexity $n^{1-1/O(q^2 \log^2 q)}$.*

We are now ready to state the following corollary, which improves on the previous best rate lower bound for relaxed LDCs [GL21] by an application of the theorem above to the setting of relaxed local decoding. This follows from the construction of a global decoder (which is able to decode the entire message) that is only guaranteed to succeed with high probability when its input is *a perfectly valid codeword*.

Corollary 7.9. *Any code $C\colon \{0,1\}^k \to \{0,1\}^n$ that is relaxed locally decodable with $q = O(1)$ queries satisfies*

$$n = k^{1+\frac{1}{O(q^2 \log^2 q)}}.$$

Proof. Let D' be the sample-based relaxed LDC with sample complexity q' obtained by Corollary 7.8 from a relaxed LDC with query complexity q for the code C. Reduce the error rate of D' to $1/3k$ by repeating the algorithm $O(\log k)$ times and taking the majority output, thus increasing the sample complexity to $O(q' \cdot \log k) = n^{1-1/t}$ with $t = O(q^2 \log^2 q)$.

Now, consider the *global decoder* G defined as follows: on input w, execute the sampling stage once and the enumeration stages of $D^w(1), \ldots, D^w(k)$ on the same samples. A union bound ensures that, with probability at least $2/3$, the outputs satisfy $D^w(i) = x_i$ for all i if $w = C(x)$.

The global decoder G obtains k bits of information from $n^{1-1/t}$ bits with probability above $1/2$. Information-theoretically, we must have

$$k \leq \frac{n^{1-1/t}}{2} = \frac{n^{\frac{t-1}{t}}}{2},$$

so that $n \geq (2k)^{1+\frac{1}{t-1}} \geq 2k^{1+\frac{1}{t-1}}$. Since $t = O(q^2 \log^2 q)$, it follows that $n = k^{1+1/O(q^2 \log^2 q)}$. □

7.3 A maximal separation between testers and proofs of proximity

Recall that a Merlin-Arthur proof of proximity (MAP, for short) for property Π is a local algorithm that receives explicit access to a proximity parameter $\varepsilon > 0$ and a purported proof string π, as well as query access to a string $x \in \Sigma^n$. It uses the information encoded in π to decide which coordinates of x to query, accepting if $x \in \Pi$ and π is a valid proof for x, and rejecting if x is ε-far from Π. In particular, a MAP with proof length 0 is simply a tester. The complexity of a MAP is defined as the sum of its proof length and query complexity. For simplicity, we consider the proximity parameter ε to be a fixed constant in the following discussion.

A *partial tester* T is a relaxation of the standard definition of a tester, that accepts inputs inside a property Π_1 and rejects inputs that are far from a *larger* property Π_2 that contains Π_1 (standard testing is the case where $\Pi_2 = \Pi_1$). We first formalise an observation made in [FGL14], which shows an equivalence between MAPs and coverings by partial testers.

Claim 7.10. *A MAP T for property $\Pi \subseteq \Sigma^n$ with proof complexity m, error rate σ and query complexity $q = q(n, \varepsilon)$ is equivalent to a collection of* partial testers $\{T_i : i \in [|\Sigma|^m]\}$. *Each $T_i(\varepsilon)$ accepts inputs in the property Π_i and rejects inputs that are ε-far from Π, with the same query complexity q and error rate σ as T. The properties Π_i satisfy $\Pi_i \subseteq B_\varepsilon(\Pi)$ and $\Pi \subseteq \cup_i \Pi_i$.*

Proof. Consider a MAP T with parameters as in the statement, and define $T_i(\varepsilon) \coloneqq T(\varepsilon, i)$ for each purported proof $i \in [|\Sigma|^m]$. Clearly the query complexity and error rate of T_i match those of T, and these testers reject points that are ε-far from Π. The property Π_i is, by definition, the set of inputs that T_i accepts (with probability at least $1 - \sigma$), which is contained in $B_\varepsilon(\Pi)$ (since elements of $\overline{B_\varepsilon(\Pi)}$ are rejected), and may possibly be empty. But since the definition of a MAP ensures that, for each $x \in \Pi$, the tester T_i^x accepts for some proof i (with probability $1 - \sigma$), we have $\Pi \subseteq \cup_i \Pi_i$.

Consider, now, a collection of testers $\{T_i : i \in [|\Sigma|^m]\}$ as in the statement, and define a MAP T that simply selects the tester indexed by the received proof string; i.e., $T(\varepsilon, i) \coloneqq T_i(\varepsilon)$. Then, with probability at least $1 - \sigma$, the MAP T rejects inputs that are ε-far from Π and accepts $x \in \Pi$ when its proof string is $i \in [|\Sigma|^m]$ such that $x \in \Pi_i$. □

As discussed in the introduction, one of the most fundamental questions regarding proofs of proximity is their relative strength in comparison to testers; that is, whether verifying a proof for an approximate decision problem can be done significantly more efficiently than solving it. This can be cast as an analogue of the P versus NP question for property testing.

Fortunately, in the setting of property testing, the problem of verification versus decision is very much tractable: one of the main results in [GR18] shows the existence of a property Π which: (1) admits a MAP with proof length $O(\log n)$ and query complexity $q = O(1)$; and (2) requires at least $n^{1-1/\Omega(q)}$ queries to be tested without access to a proof. (The lower bound of [GR18] is stated in a slightly weaker form. However, it is straightforward to see that the stronger form holds; see discussion at the end of this section.)

While this implies a nearly exponential separation between the power of testers and MAPs, it remained open whether the aforementioned sublinear lower bound on testing is an artefact of the techniques, or whether it is possible to obtain a stronger separation, where the property is harder for testers.

Claim 7.10 and Theorem 6.1 allow us to prove the following corollary, which shows that the foregoing separation is nearly tight.

Theorem 7.11. *If a property $\Pi \subseteq \Sigma^n$ admits a MAP with query complexity q, proof length m and proximity parameter $\varepsilon = \Omega(1)$, then it admits a sample-based 2ε-tester with sample complexity $m \cdot n^{1-1/O(q^2 \log^2 q)}$.*

Applying Theorem 7.11 to the special case of MAPs with *logarithmic* proof length, we obtain a sample-based tester with sample complexity $n^{1-1/O(q^2 \log^2 q)}$, showing that the separation in [GR18] is nearly optimal, and in particular that there cannot be a fully exponential separation between MAPs and testers.

Proof (of Theorem 7.11). Let Π be a property and T be a MAP with proof length m as in the statement. By Claim 7.10, there exists a collection of partial testers $\{T_i : i \leq |\Sigma|^m\}$ with query complexity q that satisfy the following. Each T_i accepts inputs in a property Π_i and rejects inputs that are ε-far from Π, with $\Pi \subseteq \cup_i \Pi_i$. By applying Corollary 7.1 to each of these testers, we obtain a collection of sample-based testers $\{S_i\}$ with sample complexity $q' = n^{1-1/O(q^2 \log^2 q)}$ for the same partial properties, but which only reject inputs that are 2ε-far from Π.

The execution of each of the S_i proceeds in two steps, as defined in Section 6.1: *sampling* (Step 1) and *enumeration* (Step 2). Note that the sampling step is exactly the same for every S_i.

Let $k = O(m \log |\Sigma|)$ such that taking the majority output from k repetitions of S_i yields an error rate of $1/(3|\Sigma|^m)$. We define a new sample-based algorithm S that repeats the following steps k times:

1. Execute both steps of S_1 (sampling and enumeration), recording the output.
2. For all $1 < i \leq |\Sigma|^m$, *only execute the enumeration step* of S_i on the samples obtained in Item 1, and record the output.

After all k iterations have finished, check if at least $k/2$ outputs of S_i were 1 for some i. If so, output 1, and output 0 otherwise.

First suppose S receives an input $x \in \Pi$, and let $i \leq |\Sigma|^m$ such that $x \in \Pi_i$. Then the majority output of the enumerations steps of S_i is 1 with probability $1 - 1/(3|\Sigma|^m) \geq 2/3$. Now suppose S receives an input x that is 2ε-far from Π. Then, for each i, the majority output of the enumeration step of S_i is 1 with probability at most $1/(3|\Sigma|^m)$. A union bound over all $i \leq |\Sigma|^m$ ensures this happens with probability at least $1/3$, in which case S correctly outputs 0.

S is therefore a 2ε-tester for the property Π with sample complexity $k \cdot q' = m \cdot n^{1-1/O(q^2 \log^2 q)}$ (recall that $\log |\Sigma| \leq \log n$), and the theorem follows. □

Interestingly, as a direct corollary of Theorem 7.11, we obtain that the general transformation in Theorem 6.1 is optimal, up to a quadratic gap in the dependency on the sample complexity, as a transformation with a smaller sample complexity could have been used to transform the MAP construction in the MAPs-vs-testers separation of [GR18], yielding a tester with query complexity that contradicts the lower bound in that result.

Theorem 7.12. *There does* not *exist a transformation that takes a robust local algorithm with query complexity q and transforms it into a* sample-based *local algorithm with sample complexity at most $n^{1-1/o(q)}$.*

Proof. Let Π be the *encoded intersecting messages* property considered in [GR18, Section 3.1], for which it was shown that Π has a MAP with query complexity q and logarithmic proof complexity, but every tester for Π requires at least $n^{1-1/\Omega(q)}$ queries. Suppose towards contradiction that a transformation as in the hypothesis exists. Then, applying the transformation to the aforementioned MAP (as in Theorem 7.11) yields a tester for Π with query complexity $n^{1-1/o(q)}$, in contradiction to the lower bound. □

On the lower bound in [GR18]. The separation between MAPs and testers in [GR18] is proved with respect to a property of strings that are encoded by relaxed LDCs; namely, the *encoded intersecting messages property*, defined as

$$\mathrm{EIM}_C = \left\{ \big(C(x), C(y)\big) \ : \ x, y \in \{0,1\}^k,\ k \in \mathbb{N} \text{ and } \exists i \in [k] \text{ s.t. } x_i \neq 0 \text{ and } y_i \neq 0 \right\},$$

where $C\colon \{0,1\}^k \to \{0,1\}^n$ is a code with linear distance, which is both a relaxed LDC and an LTC. In [GR18] it is shown that there exists a MAP with proof length $O(\log n)$ and query complexity $q = O(1)$, and crucially for us, that any tester requires $\Omega(k)$ queries to be tested without access to a proof. The best constructions of codes that satisfy the aforementioned conditions [BGH+06, CGS22, AS21] achieve blocklength $n = O(k^{1+1/q}) = k^{1+1/\Omega(q)}$, and hence the stated lower bound follows.

References

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998. 1

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *J. ACM*, 45(1):70–122, 1998. 1

[AS21] Vahid R. Asadi and Igor Shinkar. Relaxed locally correctable codes with improved parameters. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPIcs*, pages 18:1–18:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. 5, 49

[BGH+06] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Robust PCPs of proximity, shorter PCPs, and applications to coding. *SIAM J. Comput.*, 36(4):889–974, 2006. 4, 5, 7, 24, 25, 44, 49

[BGS15] Arnab Bhattacharyya, Elena Grigorescu, and Asaf Shapira. A unified framework for testing linear-invariant properties. *Random Structures & Algorithms*, 46(2):232–260, 2015. 5

[BHLM09] Eli Ben-Sasson, Prahladh Harsha, Oded Lachish, and Arie Matsliah. Sound 3-query PCPPs are long. *ACM Transactions on Computation Theory (TOCT)*, 1(2):1–49, 2009. 7

[BMR19a] Piotr Berman, Meiram Murzabulatov, and Sofya Raskhodnikova. The power and limitations of uniform samples in testing properties of figures. *Algorithmica*, 81(3):1247–1266, 2019. 5

[BMR19b] Piotr Berman, Meiram Murzabulatov, and Sofya Raskhodnikova. Testing convexity of figures under the uniform distribution. *Random Structures & Algorithms*, 54(3):413–443, 2019. 5

[BRV18] Itay Berman, Ron D. Rothblum, and Vinod Vaikuntanathan. Zero-knowledge proofs of proximity. In Anna R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, volume 94 of *LIPIcs*, pages 19:1–19:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. 6

[CFSS17] Xi Chen, Adam Freilich, Rocco A Servedio, and Timothy Sun. Sample-based high-dimensional convexity testing. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017. 5

[CG18] Clément L. Canonne and Tom Gur. An adaptivity hierarchy theorem for property testing. *Computational Complexity*, 27(4):671–716, 2018. 5

[CGdW13] Victor Chen, Elena Grigorescu, and Ronald de Wolf. Error-correcting data structures. *SIAM J. Comput.*, 42(1):84–111, 2013. 5

[CGS22] Alessandro Chiesa, Tom Gur, and Igor Shinkar. Relaxed locally correctable codes with nearly-linear block length and constant query complexity. *SIAM J. Comput.*, 51(6):1839–1865, 2022. 49

[DGG19] Irit Dinur, Oded Goldreich, and Tom Gur. Every set in P is strongly testable under a suitable encoding. In *10th Innovations in Theoretical Computer Science Conference (ITCS 2019)*, 2019. 26

[DGL21] Marcel Dall'Agnol, Tom Gur, and Oded Lachish. A structural theorem for local algorithms with applications to coding, testing, and privacy. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 1651–1665. SIAM, 2021. 1

[DGRMT22] Marcel Dall'Agnol, Tom Gur, Subhayan Roy Moulik, and Justin Thaler. Quantum proofs of proximity. *Quantum*, 6:834, October 2022. 5

[DH13] Irit Dinur and Prahladh Harsha. Composition of low-error 2-query PCPs using decodable pcps. *SIAM J. Comput.*, 42(6):2452–2486, 2013. 5

[Efr12] Klim Efremenko. 3-query locally decodable codes of subexponential length. *SIAM J. Comput.*, 41(6):1694–1703, 2012. 4

[FGL+91] Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. Approximating clique is almost NP-complete. In *Proc. 32nd IEEE Symp. on Foundations of Computer Science*, pages 2–12, 1991. 1

[FGL14] Eldar Fischer, Yonatan Goldhirsh, and Oded Lachish. Partial tests, universal tests and decomposability. In Moni Naor, editor, *Innovations in Theoretical Computer Science, ITCS'14, Princeton, NJ, USA, January 12-14, 2014*, pages 483–500. ACM, 2014. 5, 26, 47

[FLV15] Eldar Fischer, Oded Lachish, and Yadu Vasudev. Trading query complexity for sample-based testing and multi-testing scalability. In *Proceedings of the IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, 2015. 1, 4, 5, 6, 8, 10, 12, 26, 43

[GG18] Oded Goldreich and Tom Gur. Universal locally testable codes. *Chicago J. Theor. Comput. Sci.*, 2018. 25

[GGR98] Oded Goldreich, Shari Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998. 1, 4, 5, 22

[GL21] Tom Gur and Oded Lachish. On the power of relaxed local decoding algorithms. *SIAM J. Comput.*, 50(2):788–813, 2021. 1, 4, 5, 8, 10, 12, 26, 43, 46

[Gol11] Oded Goldreich. Short locally testable codes and proofs. In Oded Goldreich, editor, *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation - In Collaboration with Lidor Avigad, Mihir Bellare, Zvika Brakerski, Shafi Goldwasser, Shai Halevi, Tali Kaufman, Leonid Levin, Noam Nisan,*

Dana Ron, Madhu Sudan, Luca Trevisan, Salil Vadhan, Avi Wigderson, David Zuckerman, volume 6650 of *Lecture Notes in Computer Science*, pages 333–372. Springer, 2011. 5

[Gol17] Oded Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017. 2

[GR16] Oded Goldreich and Dana Ron. On sample-based testers. *ACM Transactions on Computation Theory (TOCT)*, 8(2):1–54, 2016. 4, 5

[GR17] Tom Gur and Ron D. Rothblum. A hierarchy theorem for interactive proofs of proximity. In Christos H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*, volume 67 of *LIPIcs*, pages 39:1–39:43. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. 5

[GR18] Tom Gur and Ron D. Rothblum. Non-interactive proofs of proximity. *Comput. Complex.*, 27(1):99–207, 2018. 6, 26, 42, 47, 48, 49

[GS06] Oded Goldreich and Madhu Sudan. Locally testable codes and PCPs of almost-linear length. *Journal of the ACM (JACM)*, 53(4):558–655, 2006. 1, 23, 26

[GS10] Oded Goldreich and Or Sheffet. On the randomness complexity of property testing. *Computational Complexity*, 19(1), 2010. 31

[GT03] Oded Goldreich and Luca Trevisan. Three theorems regarding testing graph properties. *Random Structures & Algorithms*, 23(1):23–57, 2003. 7

[HS70] András Hajnal and E. Szemerédi. Proof of a conjecture of P. Erdős. *Colloq Math Soc János Bolyai*, 4, 1970. 29

[HSX+12] Cheng Huang, Huseyin Simitci, Yikang Xu, Aaron Ogus, Brad Calder, Parikshit Gopalan, Jin Li, and Sergey Yekhanin. Erasure coding in Windows Azure storage. In *Presented as Part of the 2012 USENIX Annual Technical Conference*, pages 15–26, 2012. 4

[KR15] Yael Tauman Kalai and Ron D. Rothblum. Arguments of proximity - [extended abstract]. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 422–442. Springer, 2015. 6

[KS17] Swastik Kopparty and Shubhangi Saraf. Local testing and decoding of high-rate error-correcting codes. *Electronic Colloquium on Computational Complexity (ECCC)*, 24:126, 2017. 4

[KT00] Jonathan Katz and Luca Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC)*, 2000. 1, 4, 23

[MR10] Dana Moshkovitz and Ran Raz. Two-query PCP with subconstant error. *J. ACM*, 57(5):29:1–29:29, 2010. 5

[New91] Ilan Newman. Private vs. common random bits in communication complexity. *Information processing letters*, 39(2):67–71, 1991. 31

[RR20] Noga Ron-Zewi and Ron D. Rothblum. Local proofs approaching the witness length [extended abstract]. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 846–857. IEEE, 2020. 5

[RRR21] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. *SIAM J. Comput.*, 50(3), 2021. 6

[RS96] Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM J. Comput.*, 25(2):252–271, 1996. 1, 22

[RVW13] Guy N. Rothblum, Salil P. Vadhan, and Avi Wigderson. Interactive proofs of proximity: Delegating computation in sublinear time. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 793–802. ACM, 2013. 6

[Tre04] Luca Trevisan. Some applications of coding theory in computational complexity. *Electronic Colloquium on Computational Complexity (ECCC)*, 2004. 4

[Yek08] Sergey Yekhanin. Towards 3-query locally decodable codes of subexponential length. *J. ACM*, 55(1):1:1–1:16, 2008. 4

[Yek12] Sergey Yekhanin. Locally decodable codes. *Foundations and Trends in Theoretical Computer Science*, 6(3):139–255, 2012. 4

A Deferred proofs

In this appendix we provide the proofs of two claims and a lemma obtained from their combination, which were deferred in Section 5: Claim 5.8 provides an amplification procedure and Claim 5.9 a randomness reduction procedure for local algorithms, while Lemma 5.10 obtains both simultaneously. We remark that both claims follow from a straightforward adaptation of standard techniques, and include these proofs for completeness. We begin with the amplification procedure.

Claim A.1 (Claim 5.8 restated)**.** *Let M be a (ρ_0, ρ_1)-robust algorithm for computing $f\colon \mathcal{P} \to \{0,1\}$ (where $\mathcal{P} \subset Z \times \Sigma^n$) with error rate $\sigma \leq 1/3$, query complexity q and randomness complexity r.*

For any $\sigma' > 0$, there exists a (ρ_0, ρ_1)-robust algorithm N for computing the same function with error rate σ', query complexity $108q\log(1/\sigma')/\sigma$ and randomness complexity $108r\log(1/\sigma')/\sigma$.

Proof. Define N as the algorithm that makes $t = 108\log(1/\sigma')/\sigma$ independent runs of M and outputs the most frequent symbol, resolving ties arbitrarily. The query and randomness complexities of N clearly match the statement, and we must now prove that the error rate is indeed σ' and that N is (ρ_0, ρ_1)-robust.

Fix $z \in Z$ and $x \in \Sigma^n$ in the domain of f and let $b \coloneqq f(z,x)$. As M is ρ_b-robust at x, the algorithm satisfies $\Pr[M^y(z) = b] \geq 1 - \sigma$ for all $y \in B_{\rho_b}(x)$. By the Chernoff bound,

$$\Pr\left[M^y(z) \neq b \text{ for at least } (\sigma + 1/6)t \text{ runs}\right] \leq e^{-\frac{\sigma t}{3\cdot 36}} = e^{-\log\frac{1}{\sigma'}} < 2^{-\log\frac{1}{\sigma'}} = \sigma'.$$

The majority rule will thus yield outcome b with probability at least $1 - \sigma'$, since at least $1 - (\sigma + 1/6)t \geq t/2$ runs output b (except with probability at most σ'). As x, z and $y \in B_{\rho_b}(x)$ are arbitrary, the result follows. □

We proceed to the randomness reduction transformation.

Claim A.2 (Claim 5.9, restated)**.** *Let M be a (ρ_0, ρ_1)-robust algorithm for computing $f\colon \mathcal{P} \to \{0,1\}$ (where $\mathcal{P} \subset Z \times \Sigma^n$) with error rate σ, query complexity q and randomness complexity r.*

There exists a (ρ_0, ρ_1)-robust algorithm N for computing the same function with error rate 2σ and query complexity q, whose distribution $\tilde{\mu}^N$ has support size $3n\ln|\Sigma|/\sigma$. In particular, the randomness complexity of N is bounded by $\log(n/\sigma) + \log\log|\Sigma| + 2$.

Proof. Fix any explicit input $z \in Z$. Let $\{x_j\}$ be an enumeration of the inputs in Σ^n such that $\Pr[M^{x_j}(z) = b_j] \geq 1 - \sigma$ for some $b_j \in \{0,1\}$. Note that this includes points in the neighbourhood of a point at which M is robust which are not necessarily in the domain of f, so it suffices to show $\Pr[N^{x_j}(z) = b_j] \geq 1 - 2\sigma$ to prove the claim for N with the required query complexity and distribution.

Define the $2^r \times |\{x_j\}|$ matrix E with entries in $\{0,1\}$ as follows. Denote by $b_{ij} \in \{0,1\}$ the output of $M^{x_j}(z)$ when it executes according to the decision tree indexed by (the binary representation of) $i \in [2^r]$. Then,

$$E_{i,j} = \begin{cases} 1, & \text{if } b_{ij} \neq b_j \\ 0, & \text{otherwise.} \end{cases}$$

Note that $E_{i,j}$ simply indicates whether $M^{x_j}(z)$ outputs incorrectly on input when the outcome of the algorithm's coin flips is (the binary representation of) i. By construction, for each fixed j, a fraction of at most σ indices $i \in [2^r]$ are such that $E_{i,j} = 1$.

Let $t = 3n \ln |\Sigma|/\sigma$ and $I_1, \ldots, I_t$ be independent random variables uniformly distributed in $[2^r]$. For each fixed $j \leq |\{x_j\}| \leq |\Sigma|^n$ and $k \leq t$, we have $\mathbb{E}[E_{I_k,j}] \leq \sigma$. By the Chernoff bound,

$$\Pr\left[\sum_{k=1}^{t} E_{I_k,j} \geq 2\sigma t\right] \leq e^{-\frac{\sigma t}{3}} = e^{-n \ln |\Sigma|} < |\Sigma|^{-n}.$$

Applying the union bound over all $j \leq |\Sigma|^n$, we obtain

$$\Pr\left[\sum_{k=1}^{t} E_{I_k,j} \geq 2\sigma t \text{ for some } j\right] < 1.$$

We have thus shown, via the probabilistic method, the existence of a multi-set R_z of size $3n \ln |\Sigma|/\sigma$ such that

$$\Pr[N^{x_j}(z) \neq b_j] \leq 2\sigma,$$

where N samples its random strings uniformly from R_z (rather than from $\{0,1\}^r$), using the corresponding decision trees of M. The size of S_z is thus $\left|\tilde{\mu}^N\right| = 3n \ln |\Sigma|/\sigma$, and this sampling can be performed with $\log(n/\sigma) + \log\log|\Sigma| + 2$ random coins.

Since the decision trees of N are simply a subcollection of those of M, the query complexity of N is q and the claim follows. □

We now conclude with the following lemma, obtained by suitably combining the foregoing claims in sequence.

Lemma A.3 (Lemma 5.10, restated)**.** *Assume there exists a ρ-robust algorithm M for computing f with query complexity ℓ, error rate $1/3$ and arbitrary randomness complexity. Then there exists a ρ-robust q-local algorithm M' for f with error rate*

$$\sigma = \frac{1}{4q}$$

such that $\frac{q}{\log 8q} = O(\ell)$, or, equivalently,

$$q = O(\ell \log \ell).$$

Moreover, the distribution of M' is uniform over a multi-collection of decision trees of size $6n \ln |\Sigma|/\sigma$.

Proof. We apply both transformations in order, omitting mention of parameters that are left unchanged. Recall that M may have arbitrarily large randomness complexity.

1. Apply Claim 5.8 (error reduction) to M, obtaining M'' with error rate $\sigma'' = 1/8q$ and query complexity $q = O(\ell \log(1/\sigma'')) = O(\ell \log(8q))$ (as well as larger randomness complexity).
2. Apply Claim 5.9 (randomness reduction) to M'', thereby obtaining a new algorithm M' with error rate $\sigma = 2\sigma'' = \frac{1}{4q}$ and support size $3n \ln |\Sigma|/\sigma'' = 6n \ln |\Sigma|/\sigma$ on its distribution over decision trees. □