Electronic Colloquium on Computational Complexity, Report No. 68 (2021)

Quantum Proofs of Proximity

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

Tom Gur[*]
University of Warwick
tom.gur@warwick.ac.uk

Subhayan Roy Moulik [†]
UC Berkeley & University of Oxford
subhayan.roy.moulik@berkeley.edu

Justin Thaler[‡]
Georgetown University
justin.thaler@georgetown.edu

May 8, 2021

Abstract

We initiate the systematic study of QMA algorithms in the setting of property testing, to which we refer as *QMA proofs of proximity* (QMAPs). These are quantum query algorithms that receive explicit access to a sublinear-size untrusted proof and are required to accept inputs having a property Π and reject inputs that are ε-far from Π, while only probing a minuscule portion of their input.

Our algorithmic results include a general-purpose theorem that enables quantum speedups for testing an expressive class of properties, namely, those that are succinctly *decomposable*. Furthermore, we show quantum speedups for properties that lie outside of this family, such as graph bipartitneness.

We also investigate the complexity landscape of this model, showing that QMAPs can be *exponentially* stronger than both classical proofs of proximity and quantum testers. To this end, we extend the methodology of Blais, Brody and Matulef (Computational Complexity, 2012) to prove quantum property testing lower bounds via reductions from communication complexity, thereby resolving a problem raised by Montanaro and de Wolf (Theory of Computing, 2016).

[*]Tom Gur is supported by the UKRI Future Leaders Fellowship MR/S031545/1.

[†]SRM is supported by the National Science Foundation under the QLCI program through grant number OMA-2016245 and the Clarendon Fund.

[‡]Justin Thaler is supported in part by NSF CAREER award CCF-1845125 and DARPA under Agreement No. HR00112020022. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the United States Government or DARPA.

ISSN 1433-8092

Contents

1 Introduction

Quantum property testing is a fundamental model of sublinear-time quantum computation. Its importance stems both from the practical difficulty in manipulating large quantum states, as well as from the fertile ground that it provides for complexity theoretic investigations of the power of quantum mechanics as a computational resource. Accordingly, this model has garnered a large amount of attention in the last decade (see, e.g., [CFMDW10, HA11, ACL11, CM13, OW15, ABRW16, NV17, AA18, BOW19, GL19, BCL20], and the survey [MdW13]).

Building on the vast literature of classical property testing (cf. the recent book [Gol17]), quantum testers are defined as quantum query algorithms that solve the *approximate* decision problem of membership in a set Π (of possibly quantum objects), which is typically referred to as a *property*; that is, the tester must accept if its input is in the property Π and reject if it is *far* from Π with respect to a natural metric.

This paper is concerned with the notion of QMA, the quantum analogue of NP proofs, in property testing. Namely, we investigate the following question:

What is the power of QMA proofs for quantum property testing?

1.1 Quantum proofs of proximity

The question of decision versus verification is foundational in theoretical computer science, and extends far beyond $\mathcal{P}$ vs. $\mathcal{NP}$. Indeed, the study of *classical* proof systems in the property testing setting is well established [EKR04, BGH$^+$06, RVW13, KR15, GG16, BSCG$^+$17, GR17, BRV18, GR18, GLR18, GGR18, RRR19, RR20]. These objects are called *proofs of proximity*, and include, among others, PCPs of proximity, interactive proofs of proximity and MA proofs of proximity (MAPs). We henceforth adopt this standard terminology, noting it is synonymous with *proof systems in the property testing setting*.

A *Quantum Merlin-Arthur* (QMA) proof of proximity protocol for a property of unitaries Π, with respect to proximity parameter ε, is defined as follows. The *verifier*, a computational device given oracle access to a unitary U, receives a quantum state $|\phi\rangle$ from an all-powerful but untrusted *prover*. Making use of these two resources, it must decide whether $U \in \Pi$ or U is ε-far from Π with respect to a specific metric.[1] Such a protocol is said to verify (or *test*) Π if, with high probability, the verifier accepts in the former case and rejects in the latter (see Section 3 for a formal definition). We remark that the notion of QMA proofs of proximity is implicit in the literature as QMA query algorithms for approximate decision problems (e.g., the permutation testing problem [Aar12, ST19]). In this paper, we initialise the systematic study of the notion of QMA proofs of proximity (QMAPs), and explore its power and limitations.

The complexity of a QMAP protocol is measured with respect to the amount of resources required by the verifier. Namely, we will evaluate the efficiency of a protocol by its *proof complexity* p (the number of qubits in the proof $|\psi\rangle$) and *query complexity* q (the number of oracle calls made by the verifier for a worst-case input U). In particular, both parameters should be *sublinear* in nontrivial protocols.

Before proceeding to state our results, we briefly discuss three applications that underscore the motivation to study quantum proofs of proximity.

[1] Note that, unlike in the classical case, where Hamming distance is with few exceptions the natural choice, there are many natural metrics on the set of unitaries (e.g., those induced by the operator or Hilber-Schmidt norms).

Delegated remote quantum computation. With efficient data structures such as QRAM [GLM08] and reliable communication channels, one may envision a new paradigm of computing – where the data is stored in a trusted data centre, and can be accessed (in coherent superposition) by a remote quantum computer. This allows a $O(n)$-qubit computer to access 2^n different locations of the data in single query. This setting motivates the need for protocols for *delegated remote computation*, where a remote client interacts with a (powerful) server in order to perform computation (on remote data) that it could not perform on its own, but without trusting the server. Current experimental developments such as [MRR+14, LKS+19, DLW+21] further support this possibility [WEH18].

In this setting, suppose a client wants to compute a function f on the data, x. While the client cannot even load the entire database x, the server may compute and send $y = f(x)$ to the client and append a proof of proximity that asserts $f(x) = 1$. This enables the client to check, with sublinear resources, that x is not far from satisfying $f(x) = y$, i.e., that x is close in, say, Hamming distance to an input x' satisfying $f(x') = y$.

Quantum certification. Suppose a manufacturer produces a device that it claims implements a quantum circuit with high fidelity. However, we do not have access to the architecture or authority to crack open the device and examine it. The physical device is given to us as a black-box, into which we may feed a quantum state and receive another quantum state as output.

If we do not trust the manufacturer and would like to assert that the device is indeed implementing the claimed functionality, one alternative is to perform tomography and characterise its input-output behaviour. This needs extensive resources, however; to characterise an n-bit transformation, $2^{\Omega(n)}$ uses of the device are required [HHJ+17, OW16]. Alternatively, we could require the manufacturer to provide a QMA proof of proximity that certifies the operation of the device is at least *close* to what is expected.

The task of benchmarking and certification of quantum devices has been an prominent topic of research [EHW+20]. In fact, the idea of using tools from property testing to this end has been suggested in past works [Che00, HM13, HLM17, BOW19] and is closely related to self-testing [TKV+18, RKB18, ŠB19].

Complexity Class Separations in Property Testing. A fundamental question in quantum complexity is to determine whether $\mathcal{QMA}$ (the quantum analog of $\mathcal{NP}$) is strictly more powerful than $\mathcal{BQP}$ (the quantum analog of $\mathcal{P}$). Quantum proofs of proximity offer a natural setting to study analogous questions. As we explain shortly, we show unconditionally that in the property testing setting, $\mathcal{QMA}$ protocols are exponentially more powerful than both $\mathcal{BQP}$ and (classical) $\mathcal{MA}$ protocols for certain problems. and thus that QMA proofs of proximity are "larger than the sum of their parts" (see Section 1.2.1). We also show that they nevertheless have limited power, and investigate the complexity landscape surrounding QMAPs (see Section 1.2.2).

1.2 Our results

Our main results are divided into two parts: Section 1.2.1 covers algorithmic results, where we show sufficient conditions for properties to admit efficient QMAP protocols, while Section 1.2.2 charts fundamental aspects of the complexity landscape surrounding quantum proofs of proximity.

We write $\mathcal{QMAP}(\varepsilon, p, q)$ for the class of ε-testable properties by a QMA proof of proximity protocol with proof length p and query complexity q (non-calligraphic acronyms to refer to algorithms and protocols, while calligraphic letters denote complexity classes).

1.2.1 Algorithms

We show two general classes of properties whose structure allows for efficient QMAP (QMA proof of proximity) protocols. Moreover, these protocols only require *classical* proofs (though the verifier is quantum). The first class comprises of what we call *decomposable properties*, which generalise the "parametrised concatenation properties" introduced in [GR18].

Roughly speaking, a property Π is (k, s)-decomposable if testing whether $x \in \Pi$ can be reduced, with the help of a message of length s from the prover, to that of testing whether $x^{(i)} \in \Lambda^{(i)}$ for k smaller strings $x^{(i)}$ and properties $\Lambda^{(i)}$ (see Definition 3). Since there may be several decompositions of the same string, the prover's message is said to *specify* a decomposition, i.e., mappings $x \mapsto x^{(i)}$ and $\Pi \mapsto \Lambda^{(i)}$.

Theorem 1 (Theorem 4.1, informally stated)**.** *If a property Π is (k, s)-decomposable into strings of length m, each of which is ε-testable by a MAP protocol with proof complexity p and query complexity $q = q(m, \varepsilon)$, then*

$$\Pi \in \mathcal{QMAP}(\varepsilon, s + kp, \tilde{O}(q)) \; .$$

The second class of properties amenable to QMA proofs of proximity are those admitting (classical) MAPs which do not receive a proximity parameter explicitly, but rather reject strings ε-far from the property with probability that is a function of ε. Such algorithms are called *proximity-oblivious* MAPs and readily admit quantum speedups (via the technique of amplitude amplification; see Section 3.2).

Theorem 2. *If a property Π admits a proximity-oblivious MAP protocol with proof complexity p and query complexity q, which always accepts $x \in \Pi$ and rejects when x is ε-far from Π with probability $\rho(\varepsilon) > 0$, then*

$$\Pi \in \mathcal{QMAP}\left(\varepsilon, p, O\left(\frac{q}{\sqrt{\rho(\varepsilon)}}\right)\right) \; .$$

As applications of Theorem 1, we show:

Corollary 1. *Let $\Pi_{k,[n]}$ denote the set of k-*monotone *functions $f \colon [n] \to \{0, 1\}$, i.e., those which change from nondecreasing to nonincreasing and vice-versa at most $k - 1$ times. For all $\varepsilon \in (0, 1)$,*

$$\Pi_{k,[n]} \in \mathcal{QMAP}\left(\varepsilon, k \log n, \tilde{O}\left(\frac{1}{\sqrt{\varepsilon}}\right)\right) \; .$$

Corollary 2 (Corollary 7, informally stated)**.** *For every $k \in [n]$, the property of Eulerian graph orientations of $K_{2,n-2}$ is ε-testable by a QMAP protocol with proof complexity $\tilde{O}(k)$ and query complexity $\tilde{O}\left(\frac{n}{k\sqrt{\varepsilon}}\right)$.*

Applying Theorem 2, we also show the following:

Corollary 3 (Corollaries 5 and 6, informally stated)**.** *For every $k \in [n]$, acceptance by read-once branching programs and membership in context-free languages are both ε-testable by QMAP protocols with proof complexity $\tilde{O}(k)$ and query complexity $O\left(\frac{n}{k\sqrt{\varepsilon}}\right)$.*

Corollaries 2 and 3, achieve a dependence on ε that is quadratically better as compared to the best known MAPs [GR18, GGR18], while Corollary 1 is more efficient than the best known (classical) testers and MAPs for a wide range of parameters (see Section 4.2).

Classically, casting *exact decision* problems in the framework of proofs of proximity (i.e., testing with respect to proximity parameter $\varepsilon = 1/n$) is completely trivial except for degenerate cases, as most functions of sublinear query complexity are extremely simple. Rather surprisingly, this is *not* the case quantumly, and indeed setting $\varepsilon = 1/n$ in Corollaries 1 to 3 yields sublinear algorithms for the corresponding exact decision problems. For *layered* branching programs, we also prove the following, which improves on the parameters of Corollary 3 and lifts the read-once restriction:

Theorem 3. *There exists a QMA protocol for acceptance of n-bit strings by layered branching programs of width $w = w(n)$ and length $\ell = \ell(n)$ with query complexity $O(\ell^{2/3})$ and proof complexity $O(\ell^{2/3} \log w)$.*

For details on decomposability and its implications, see Section 4. Finally, we prove that QMAP protocols are useful beyond proximity-oblivious and decomposable properties. The problem of testing bipartiteness of a graph does not fit either class, yet admits an efficient protocol nonetheless (see Section 5).

Theorem 4 (Theorem 5.1, informally stated)**.** *Graph bipartiteness (in the bounded-degree model) is ε-testable by a QMAP protocol with proof complexity $\tilde{O}(\sqrt{n})$ and query complexity $\tilde{O}(n^{1/3}/\varepsilon^{4/3})$.*

1.2.2 Complexity separations

Our second collection of results aims to chart the complexity landscape of quantum proofs of proximity. Recall that $\mathcal{QMAP}(\varepsilon, p, q)$ is the class of ε-testable properties by a QMAP with proof complexity p and query complexity q. The classes $\mathcal{MAP}$ (MA proofs of proximity) and $\mathcal{IPP}$ (interactive proofs of proximity) are defined analogously. $\mathcal{PT}(\varepsilon, q)$ and $\mathcal{QPT}(\varepsilon, q)$ are the properties admitting classical and quantum ε-testers with query complexity q, respectively, and $\mathcal{QCMAP}(\varepsilon, p, q)$ is the restriction of $\mathcal{QMAP}(\varepsilon, p, q)$ where the proofs are classical. Formal definitions of all of these classes can be found in Section 2 and Section 3.1.

We write complexity classes with the parameters omitted (e.g., $\mathcal{QMAP}$) to denote the corresponding class of properties such that for some proximity parameter $\varepsilon \in (0, 1)$ that is a universal constant, there is a protocol with proof and query complexities bounded by polylog(n).

Our first result here shows the existence of a property that admits efficient QMAPs, yet neither quantum property testers nor MAPs can efficiently test the property.

Theorem 5. *There exists a property Π such that, for any small enough constant $\varepsilon > 0$,*

$$\Pi \in \mathcal{QMAP}(\varepsilon, \log n, O(1)) \text{ and}$$
$$\Pi \notin \mathcal{QPT}(\varepsilon, o(n^{0.49})) \cup \mathcal{MAP}(\varepsilon, p, q)$$

when $p \cdot q = o(n^{1/4})$. In particular,

$$\mathcal{QMAP} \not\subseteq \mathcal{QPT} \cup \mathcal{MAP} .$$

Theorem 5 is, in fact, implied by a stronger result. We show that, for certain properties, MAPs are stronger than quantum testers, so $\mathcal{MAP} \not\subseteq \mathcal{QPT}$ (Theorem 6.3); and, for others, quantum

$$\begin{tikzcd}
\mathcal{IPP} & & \mathcal{QMAP} \arrow[ll, "\text{Theorem B.2, implied by [GLR18, ST19]}"'] \\
& & \mathcal{QCMAP} \arrow[u, "\text{Theorem A.1, implicit in [AK07]}"'] \\
\mathcal{MAP} \arrow[uu, "\text{[GR18]}"] \arrow[urr] \arrow[rr, leftrightarrow, "\text{Theorem 5}"] & & \mathcal{QPT} \arrow[u] \\
& \mathcal{PT} \arrow[ul, "\text{[GR18]}"] \arrow[ur, "\text{[BFNR08]}"'] &
\end{tikzcd}$$

Figure 1: Classification of complexity classes. An arrow from $\mathcal{A}$ to $\mathcal{B}$ is present when there exists a property requiring $n^{\Omega(1)}$ proof length or query complexity by algorithms of $\mathcal{A}$ but only $\operatorname{polylog} n$ proof/query complexity by algorithms of $\mathcal{B}$ (with respect to a proximity parameter $\varepsilon = \Omega(1)$ that is a universal constant). The grey arrows are previously known separations.

testers are stronger than MAPs, so $\mathcal{QPT} \not\subseteq \mathcal{MAP}$ (Theorem 6.4). Combining these results, we conclude that $\mathcal{QCMAP} \not\subseteq \mathcal{QPT} \cup \mathcal{MAP}$, i.e., even QMAPs with classical proofs suffice to obtain an exponential speedup over both MAPs and quantum testers.

Next, we establish *limitations* on the algorithmic power of QMAPs, showing that there exist explicit properties that are extremely difficult for QMAPs. First, we observe that known lower bounds on the complexity of $\mathcal{QMA}$ protocols for the *Permutation Testing* problem [Aar12, ST19] yield an explicit property that does not have any $\mathcal{QMA}$ protocol of polylogarithmic proof length and query complexity, and in fact establishes that $\mathcal{QMAP} \not\subseteq \mathcal{IPP}$ (see Appendices A and B for details). Next, we establish an entire class of properties that cannot be solved by efficient $\mathcal{QMAP}$ protocols. This extends and simplifies one of the main results of [FGL14], which obtains the same result, but for classical MAPs.

Theorem 6 (Corollary 9, informally stated)**.** *If a non-trivial property* Π *is* k*-wise independent and* $\varepsilon = \Omega(1)$ *is sufficiently small, then* $\Pi \notin \mathcal{QMAP}(\varepsilon, p, q)$ *when* $pq = o(k)$*.*

Finally, we observe that a straightforward adaptation of a known result yields an additional separation: The $\mathcal{QMA}$ vs. $\mathcal{QCMA}$ oracle separation of Aaronson and Kuperberg [AK07] carries over to the property testing setting, implying $\mathcal{QMAP} \not\subseteq \mathcal{QCMAP}$. We thus obtain the complexity landscape shown in Fig. 1.

1.3 Technical overview

In this section, we discuss the high-level ideas of the techniques used in the proofs of the results stated in Section 1.2. Our discussion is divided into algorithmic techniques and lower bounds.

In Section 1.3.1 we show how to construct quantum proofs of proximity for properties that can be decomposed into sub-problems, and we prove that these QMAP protocols outperform both quantum testers as well as classical proof of proximity protocols. Moreover, we give an overview of a construction of an efficient QMAP for a natural property of bounded-degree graphs, *bipartiteness*, which does not fall into the decomposability paradigm.

In Section 1.3.2, we introduce some of the lower bound techniques that we use in charting the complexity landscape of quantum proofs of proximity. En route, we extend the framework of Blais, Brody and Matulef [BBM12] to show lower bounds for *quantum* property testers. To the best of our knowledge, this is the first quantum testing lower bound proved via a reduction from quantum communication complexity, an open problem raised by Montanaro and de Wolf [MdW13]. In addition, we show how to prove lower bounds on QMAP algorithms via an argument about the threshold degree of Boolean functions.

1.3.1 Algorithmic techniques

As a warm-up, consider the *exact decision* problem of verifying that an n-bit string x has even parity. This is maximally hard for both MA algorithms and quantum query algorithms, requiring $\Omega(n)$ queries to the bit string, and thus asymptotically no better than trivially querying every coordinate. As we will see next, however, QMA algorithms can capitalise on having a proof *and* quantum processing power to break the linear barrier.

We rely on the technique of *amplitude amplification* [BHMT02] to obtain such an algorithm with sublinear proof and query complexities. Loosely speaking, amplitude amplification takes a (randomised) decision algorithm that always accepts yes-inputs and rejects no-inputs with probability ρ, and produces an algorithm with rejection probability $2/3$ (for no-inputs) using the former algorithm only $O(1/\sqrt{\rho})$ times as a subroutine.

We can thus obtain a QMA (query) algorithm for the parity problem as follows. The proof string specifies the purported parities of each block of an equipartition of the input $x \in \{0,1\}^n$ into p blocks of length n/p. The verifier first checks that the proof string has even parity, rejecting immediately otherwise. Then, the verifier performs amplitude amplification on the following subroutine: sample $i \in [p]$ uniformly at random, read the entire block of n/p bits and check that its parity coincides with that claimed by the proof; if so, accept, and reject otherwise.

Note that the aforementioned subroutine always accepts if x has even parity and the proof corresponds to the parity of every block. On the other hand, if a string has odd parity and the proof has even parity, *at least one bit of the proof* disagrees with the corresponding block, so that the subroutine rejects with probability at least $1/p$. Since we need only repeat $O(\sqrt{p})$ times, each of which queries n/p bits, the query complexity of our algorithm is $q = O(n/\sqrt{p})$; in particular, if $p = n^{2/3}$ then $q = O(n^{2/3})$.[2]

This is a special case of a more general phenomenon, which holds for all *decomposable* properties (see Section 4.3 for a discussion of how exact decision follows as a special case). Since amplitude amplification can only be applied to one-sided algorithms (i.e., those that always accept a valid input), we restrict our attention to this type of algorithm hereafter.

[2]Subsequent to discovering the QMA protocol for parity, we learned that the same result was obtained in unpublished prior work of Alessandro Cosentino, Robin Kothari, and John Watrous.

Decomposable properties. Roughly speaking, a property Π is said to be (k,s)-decomposable if a "specification" of length s efficiently reduces testing Π to testing k smaller properties $\Lambda^{(1)},\dots,\Lambda^{(k)}$. More precisely, Π is (k,s)-decomposable if: (1) there exists an s-bit string that specifies a set of k properties $\Lambda^{(i)}$ as well as k strings $x^{(i)} \in \{0,1\}^{m_i}$ whose bits are determined by a small number of bits of x; and (2) ε-testing $x \in \{0,1\}^n$ with respect to Π reduces to testing $x^{(i)}$ with respect to $\Lambda^{(i)}$ in the following sense: when $x \in \Pi$ then $x^{(i)} \in \Lambda^{(i)}$ for all $i \in [k]$, whereas when x is ε-far from Π, then $x^{(i)}$ is ε_i-far from $\Lambda^{(i)}$ for some ε_i satisfying $\mathbb{E}_i[\varepsilon_i] = \Omega(\varepsilon)$, where the expectation over i means that i is sampled with probability proportional to m_i. If the specification is short (i.e., $s = O(k\log n)$), we say Π is *succinctly* k-decomposable (see Section 4 for details). Decomposable properties generalise the notion of *parametrised k-concatenation properties* introduced in [GR18], which corresponds to the special case of a $(k,0)$-decomposition that is an equipartition of the input string.

Our simplest example of a decomposable problem is that of testing the set of k-monotone functions $f : [n] \to \{0,1\}$, i.e., functions that change from non-decreasing to non-increasing and vice-versa at most $k-1$ times. A natural decomposition of this property is to specify the set of at most $k-1$ "critical points", which induce a set of at most k subfunctions f_i that are monotone and overlap with f_{i-1} and f_{i+1} at their endpoints; then, it suffices to test for (1-)monotonicity of each subfunction. More precisely, this property is $(k,(k-1)\log n)$-decomposable (thus succinctly k-decomposable), and given the (alleged) critical points $n_1 < n_2 < \cdots < n_{k-1}$, the subproperty $\Lambda^{(i)}$ for odd (resp. even) i is the set of non-decreasing (resp. non-increasing) functions on $[m_i]$, where $m_i = n_i - n_{i-1} + 1$ (with $n_0 = 1$ and $n_k = n$). Note, moreover, that if f is ε-far from k-monotone, then its absolute distance from all functions specified by the critical points is at least εn; thus, denoting by ε_i the distance of f_i to $\Lambda^{(i)}$, we have $\sum_i \varepsilon_i m_i \geq \varepsilon n$, implying $\mathbb{E}_i[\varepsilon_i] = \Omega(\varepsilon)$. We remark that decomposing other properties (e.g., branching programs, context-free languages, and Eulerian graph orientations) is much less straightforward and often allows for breaking the property into any desired number of sub-properties, which in turn admits proof length versus query complexity tradeoffs. See Section 4 for details.

Given a (k,s)-decomposable property that admits MAPs for the subproperties $\Lambda^{(i)}$, a natural protocol for Π is to sample $i \in [k]$ uniformly at random and execute the verifier for $\Lambda^{(i)}$. Note that, if these MAPs have proof complexity p and query complexity q, the protocol for Π has proof length $s + kp$. Moreover, $\mathbb{E}_i[\varepsilon_i] = \Omega(\varepsilon)$ means that a randomly chosen $i \in [k]$ is (in expectation) at distance roughly ε from $\Lambda^{(i)}$, so it is reasonable to expect that $O(1/\varepsilon)$ classical repetitions of the base protocol would ensure a rejection with high probability, and that a QMAP protocol can make do with only $O(1/\sqrt{\varepsilon})$ repetitions using amplitude amplification.

The above outline glosses over the fact that we have no information on *the distribution of errors* $(\varepsilon_1,\dots,\varepsilon_k)$. For example, it may be that most ε_i are of the same order of magnitude (in which case a random $i \in [k]$ is likely to point to a mildly corrupted $x^{(i)}$), or it may be that a few ε_i are very large while all other ε_i are small or even zero (in which case $x^{(i)}$ is unlikely to be corrupted for a random $i \in [k]$, but when it is, the amount of corruption is large). Fortunately, this issue can be addressed by the technique of *precision sampling* [Lev85], incurring a merely logarithmic overhead. We thus obtain a QMAP protocol for ε-testing Π with proof complexity $s + kp$ and query complexity $\tilde{O}(q)$ (see Theorem 4.1 for details).

Bipartiteness testing. Consider the problem of testing whether a bounded-degree graph G (given as an oracle to its adjacency list) is bipartite or far from any bipartite graph. (Note that this is not a decomposable property.) There exists a MAP protocol for a promise variant of this problem, where graphs are *rapidly-mixing* [GR18]. We will show that, although mild quantum speedups can be obtained by an immediate application of amplitude amplification or by replacing a classical subroutine with a more efficient quantum analogue, taking full advantage of the available quantum resources requires new ideas.

Let us first consider the (classical) MAP verifier for bipartiteness, which receives a subset of vertices S of size k, allegedly on the same side of a bipartition, as a proof. To test with respect to proximity parameter ε, the verifier repeats the following procedure: sample a uniform random vertex v, take roughly $n/(k\varepsilon)$ short (lazy) random walks starting from v, recording whether the walk ended at a vertex in S as well as the parity of the walk (i.e., the parity of the number of non-lazy steps). If two walks start from the same vertex v and end in S with different parities, then reject; otherwise, accept. Setting $m \coloneqq n/k$, the query complexity of (one iteration of) the verifier is m/ε (ignoring constants and polylogarithmic factors).

If the graph is bipartite and the proof S is indeed on the same side of a bipartition, there cannot exist two paths from the same vertex into S with different parities (as that would imply a path of odd length with both endpoints on the same side). Therefore, the verifier always accepts in this case. If the graph is ε-far from bipartite, however, each iteration finds evidence to this effect with probability $\Omega(\varepsilon)$. Thus, the classical verifier samples a new vertex roughly $1/\varepsilon$ times, for a total query complexity of m/ε^2.

Now, one immediate way to improve this algorithm is to perform amplitude amplification: the resulting algorithm repeats the procedure $1/\sqrt{\varepsilon}$ times, improving the query complexity to $m/\varepsilon^{3/2}$. A second (and less straightforward) strategy is to use the quantum *collision-finding* algorithm [Amb07] to reduce the number of random walks taken from each vertex, as in [ACL11], to $(m/\varepsilon)^{2/3}$.[3] This strategy reduces the required number of queries to $m^{2/3}/\varepsilon^{5/3}$, improving the dependency on m but achieving a worse one on ε.

Of course, this begs the question: why not apply both optimisations? Alas, this is not possible: amplitude amplification does not apply to quantum algorithms that make intermediate measurements,[4] which is the case for collision-finding. It may seem, then, that good improvements in the dependency on m and ε are mutually exclusive.

It is possible, however, to obtain a better dependency on ε without sacrificing the dependency on m at all. The high-level idea towards accomplishing this is to incorporate the set of sampled vertices into the subroutine sped up by the collision-finding algorithm, thereby reducing the number of samples to $1/\varepsilon^{2/3}$ and thus the query complexity to $m^{2/3}/\varepsilon^{4/3}$.

More precisely, fix a vertex v and let g_v denote the mapping $r \mapsto (a, b) \in \{0,1\}^2$ obtained by executing a random walk starting from v with r as its inner randomness, where $a = 1$ if the walk stops at a vertex in S and b is the parity of the walk. The collision-finding algorithm is capable of finding a pair r_0, r_1 such that $g_v(r_i) = (1, i)$ for $i \in \{0,1\}$, if such a pair exists. The query complexity of the collision-finding algorithm is the domain size to a 2/3 power, and, since we take m/ε walks from v, the number of queries is roughly $(m/\varepsilon)^{2/3}$. Next, suppose we sample $1/\varepsilon$ vertices v, and consider the mapping $(v, r) \mapsto (v, g_v(r))$. If we use the collision-finding

[3]Here and throughout, we use the term collision-finding to refer to Ambainis' algorithm that, for any f with 1-certificate complexity at most 2, uses $\Theta(n^{2/3})$ queries and with constant probability outputs a 1-certificate when run on any input $x \in f^{-1}(1)$.

[4]Amplitude amplification requires that the algorithm be *invertible*, i.e., be given by a unitary A, as the technique repeatedly applies A and A^{-1}. While a classical (decision) algorithm can be made reversible, thus invertible, by standard procedures, this is not the case for quantum algorithms with intermediate measurements.

algorithm to look for collisions of the type $(v, r_0), (v, r_1)$ such that $(v, r_i) \mapsto (v, 1, i)$ for $i \in \{0, 1\}$, we obtain a QMAP protocol for bipartiteness with proof length $O(k \log n)$ and query complexity $\tilde{O}((m/\varepsilon^2)^{2/3}) = O((n/k)^{2/3}/\varepsilon^{4/3})$ (see Theorem 5.1 for details).

1.3.2 Lower bounds

In this section, we highlight two techniques that we exploit to prove complexity separations and limitations on QMAPs: (1) proving quantum testing lower bounds via reductions from quantum communication complexity [BBM12], which we use to show a separation between MAPs and quantum testers; and (2) proving QMAP lower bounds by studying the threshold degree of Boolean functions.

Quantum testing lower bounds via reductions from communication complexity. The methodology of [BBM12] has proven very successful for showing *classical* property testing lower bounds. However, extending this methodology to the quantum setting poses an inherent difficulty that we expand upon next. Following the exposition of [MdW13], we illustrate the methodology and the difficulty in the quantum setting by considering the problem of testing whether a function $f\colon \{0,1\}^n \to \{0,1\}$ is k-linear, i.e., a Fourier character of weight k.

We can obtain query complexity lower bounds on testers via a reduction from the randomised *communication complexity* problem of disjointness, as follows. Recall that, in the disjointness problem, Alice receives $x \in \{0,1\}^n$ and Bob receives $y \in \{0,1\}^n$ (for lower bound purposes, we may assume without loss of generality that both bit strings are promised to have Hamming weight $k/2$ for some known $k \in [n]$), and their goal is to decide whether or not there exists an index $i \in [k]$ such that $x_i = y_i = 1$, while communicating a minimal number of bits.

Suppose that there exists a property tester for k-linearity with query complexity q. We will use this tester to construct a communication complexity protocol for disjointness (i.e., deciding if, for every $i \in [n]$, either $x_i = 0$ or $y_i = 0$) as follows. First, Alice and Bob use shared randomness and simulate the tester on the input f, interpreted as a function mapping $\{0,1\}^n$ to $\{0,1\}$ defined as $f(z) = \bigoplus_{i\in[n]} z_i \cdot (x_i \oplus y_i)$. To simulate a query $f(z)$, Alice computes $a(z) = \bigoplus_{i\in[n]} z_i \cdot x_i$ and sends it to Bob, while Bob computes $b(z) = \bigoplus_{i\in[n]} z_i \cdot y_i$ and sends it to Alice. Since $f(z) = a(z) \oplus b(z)$, each query to f incurs 2 bits of communication. Moreover, if x and y are disjoint, then f is k-linear; and if they are not disjoint, f is ℓ-linear for some $\ell < k$, and is in particular $1/2$-far from every k-linear function. Therefore, the simulated tester indeed solves the communication problem, so that the $\Omega(k)$ lower bound for the latter implies an $\Omega(k)$ lower bound for testing k-linearity.

An attempt to extend this to *quantum* testers, however, reveals a severe bottleneck in the reduction. Note that, classically, the fact that Alice and Bob can use shared randomness to fix a *deterministic* tester to simulate is crucial: at every step, both parties know which query the tester will make next *without* the need to communicate it. The problem is that there is no way to fix the "quantumness" using shared randomness. Details follow.

While disjointness is still hard in the quantum communication complexity model, communicating the query (which may be in a superposition) that the quantum tester requires will incur a *linear* overhead, rendering the reduction useless. Namely, to simulate a query to f *in superposition*, the parties need to exchange all n qubits at each round: Alice would apply the unitary (on $n+1$ qubits) $|z\rangle\,|b\rangle \mapsto |z\rangle\,|b \oplus a(z)\rangle$, and send the $(n+1)$-qubit state to Bob, who applies $|z\rangle\,|b \oplus a(z)\rangle \mapsto |z\rangle\,|b \oplus a(z) \oplus b(z)\rangle = |z\rangle\,|b \oplus f(z)\rangle$ and returns them to Alice. Simulating a single query then requires the communication of $2n+2$ qubits, rather than the 2 needed by a classical tester. Thus, the reduction can only prove a degenerate $\Omega(1)$ testing lower bound. This is, in fact, not suprising, since k-linearity is testable with $O(1)$ queries by the Bernstein-Vazirani [BFNR08] algorithm!

To make the bottleneck explicit and show how this barrier can be broken, we take a coding-theoretic perspective similar to [Gol20]. First, note that testing k-linearity is a special case of *testing a subset of a code*: namely, k-linear functions correspond to the *Hadamard encoding* of strings with Hamming weight k. Note that the aforementioned quantum simulation strategy communicates a representation of the encoding that is related *logarithmically* to the blocklength; however, the exponential blocklength of the Hadamard encoding means such "compressibility" is canceled out.

This is not the case in general: for a linear code $C\colon \{0,1\}^n \to \{0,1\}^{n'}$ with $n' = \text{poly}(n)$, only $\log n' = O(\log n)$ qubits are necessary to represent $C(x)$ and $C(y)$. More precisely, Alice can apply the $O(\log n)$-qubit unitary $|i\rangle\,|b\rangle \mapsto |i\rangle\,|b \oplus C(x)_i\rangle$ and send all $O(\log n)$ qubits to Bob, who applies $|i\rangle\,|b\rangle \mapsto |i\rangle\,|b \oplus C(y)_i\rangle$ and returns them. This composition of unitaries is

$$|i\rangle\,|b\rangle \mapsto |i\rangle\,|b \oplus C(x)_i \oplus C(y)_i\rangle = |i\rangle\,|b \oplus C(x \oplus y)_i\rangle \ ,$$

simulating a query with *logarithmic*, rather than linear, overhead.

Therefore, the $\Omega(\sqrt{n})$ quantum communication lower bound for disjointness [Raz03] implies an $(n')^{\Omega(1)}$ lower bound for the problem of testing a subset of C. Indeed, we show that, for a linear code $C\colon \mathbb{F}^n \to \mathbb{F}^{n'}$ (over a larger field), the property $\{C(z) : z \in \{0,1\}^n\}$, of *Booleanity*,[5] which may be of interest in PCP constructions, has a testing lower bound of $\Omega(\sqrt{n}/\log n)$ via a reduction from disjointness (see Section 6.2 for details). We remark that since this technique is used to show a separation between quantum testers and MA proofs of proximity, we use codes that are *locally testable* and *relaxed locally decodable*, which allow for efficient testing by a MAP. Since there exist such codes with a nearly-linear blocklength [BGH+06], the lower bound we obtain is only slightly worse than a square root.

QMAP lower bounds via threshold degree. We prove lower bounds for QMAPs via the *threshold degree* of related functions. A function $f\colon \{0,1\}^n \to \{0,1\}$ is said to have threshold degree (at most) d if there exists a degree-d polynomial $P(X_1, \ldots, X_n)$ over $\mathbb{R}$ such that $f(x) = 1$ if $P(x) > 0$ and $f(x) = 0$ if $P(x) < 0$; in other words, the threshold degree of f is the smallest degree of a polynomial that *sign-represents* f.

As a first step, we show that the inclusion $\mathcal{QMA} \subseteq \mathcal{PP}$ [MW05] (in the *polynomial-time* setting) carries over to the property testing setting, implying $\mathcal{QMAP} \subseteq \mathcal{UPP}$.[6] Next, we show that the query complexity of a UPP algorithm that computes f is exactly the threshold degree of f (this result is folklore, we provide a proof for completeness). Since a property Π induces the (partial) function f_Π such that $f_\Pi(x) = 1$ when $x \in \Pi$ and $f_\Pi(x) = 0$ when x is ε-far from Π, the query complexity of a UPP algorithm that "tests" Π (i.e., computes f_Π) is a lower bound on the product pq of the proof and query complexities of any QMAP protocol for testing Π. Finally, we show that if Π is *k-wise independent* (i.e., looks perfectly random on any susbset of k coordinates) and not too large, the threshold degree of f_Π is at least k, so that $pq = \Omega(k)$ (see Section 7 for details).

[5]In fact, we show (and use to prove the separation $\mathcal{QPT} \not\subseteq \mathcal{MAP}$) a lower bound for *non*-Booleanity; but the symmetry of the model of communcation complexity implies the same lower bound holds for Booleanity as well.

[6]$\mathcal{UPP}$ is the query model version of the class $\mathcal{PP}$ of unbounded-error randomised algorithms, where in particular the amount of randomness available to the algorithm is unbounded. Since PP algorithms run in polynomial time, they may access at most a polynomial number of random bits; this restriction does not hold for $\mathcal{UPP}$.

In particular, any code with linear dual distance and small enough rate is an example of a hard property for QMAPs, requiring proof and query complexities that satisfy $pq = \Omega(n)$ for proximity parameter $\varepsilon = \Omega(1)$.

1.4 Open problems

This work begins the exploration of quantum proofs of proximity, leaving a host of uncharted research directions. We wish to highlight a small number of open problems, which we find to be of particular interest.

Given our focus on quantum MA (i.e., *non-interactive*) proofs of proximity, it is natural to ask what is achievable by allowing quantum property testers to interact with quantum provers, as opposed to static proofs.

Open question 1. What is the power of quantum IP proofs if proximity (QIPPs)?

More specifically, it is known that there exist classical interactive proof of proximity (IPP) protocols with $\tilde{O}(\sqrt{n})$ proof and query complexities for large classes of languages [RVW13, RR20]. Moreover, these complexities are optimal (up to polylogarithmic factors) for *classical* protocols, under reasonable cryptographic assumptions [KR15]. Could quantum interactive proofs break the square-root barrier?

Open question 2. Can QIPPs test logspace-uniform $\mathcal{NC}$ languages with $o(\sqrt{n})$ proof and query complexities?

Finally, while we show a strong lower bound for QMAPs for k-wise independent properties, they do not rule out the existence of sublinear QMAP protocols. Could a stronger lower bound be shown?

Open question 3. Do there exist maximally hard properties for QMAPs, requiring $\Omega(n)$ proof and query complexities?

Organisation

The rest of this paper is organised as follows. In Section 2, we discuss the preliminaries for the technical sections. In Section 3, we formally define QMA proofs of proximity and show they enable speedups for proximity-oblivious MAPs. In Section 4, we define decomposability and prove the bulk of our algorithmic results, including exact decision problems as a special case. We show a QMAP protocol for testing graph bipartiteness in Section 5, and prove our separation results in Section 6 (which Appendices A and B complement with separations implied by known results). Finally, in Section 7 we prove lower bounds for QMAPs.

2 Preliminaries

We begin with standard notation. For an integer $\ell \geq 1$, we denote by $[\ell]$ the set $\{1, 2, \ldots, \ell\}$. We use $\mathrm{polylog}(n)$ to denote an arbitrary polylogarithmic function, i.e., a polynomial in the logarithm of n. For ease of notation, we also define $N := 2^n$.

We use $\mathcal{H}, \mathcal{K}$ to denote arbitrary finite-dimensional Hilbert spaces and use indices to differentiate between distinct spaces. The set of linear operators mapping $\mathcal{H}$ to $\mathcal{K}$ is denoted by $\mathcal{L}(\mathcal{H}, \mathcal{K})$; the shorthand $\mathcal{L}(\mathcal{H})$ stands for $\mathcal{L}(\mathcal{H},\mathcal{H})$. The set of positive semidefinite operators on $\mathcal{H}$ having unit trace is denoted by $\mathrm{pos}(\mathcal{H})$. The set $T(\mathcal{H},\mathcal{K})$ consists of the linear mappings from $\mathcal{L}(\mathcal{H})$ to $\mathcal{L}(\mathcal{K})$. We say T is a completely positive map (CP-Map) if $T \otimes I_{\mathcal{L}(\mathcal{H})}$ is positive for all $\mathcal{H}$, where $I_{\mathcal{K}}$ denotes the identity operator on a Hilbert space $\mathcal{K}$. Furthermore, T is a completely positive trace preserving map (CPTP map) if T is a trace preserving CP-Map, i.e., such that $\mathrm{Tr}(T(\rho)) = \mathrm{Tr}(\rho)$ for all ρ. $\mathcal{U}(\mathcal{H})$ denotes the set of unitary operators on $\mathcal{H}$. For a unitary transformation $U \in \mathcal{U}$, its conjugate transpose is denoted $U^\dagger$.

A pure state is a unit vector in the Hilbert space $\mathcal{H}$, and represented by the Dirac notation, e.g., $|\psi\rangle$. A mixed state is a distribution on pure states $\{p_k, |\psi_k\rangle\}$, represented as a *density matrix* $\rho = \sum p_k |\psi_k\rangle\langle\psi_k|$. We write $\dim(\mathcal{H})$ to denote the dimension of the Hilbert space $\mathcal{H}$. Also, for brevity, we represent $|0\rangle^{\otimes n}$ as $|\mathbf{0}\rangle$, for an arbitrary $n \geq 1$, where $|0\rangle = [1 \;\; 0]^T$.

Complexity classes. We use the convention of writing complexity classes in calligraphic capitals and denoting algorithms or protocols by latin capitals, e.g., $\mathcal{BQP}$ is a complexity class whose problems are solvable by BQP algorithms and $\mathcal{IP}$ is a complexity class characterised by IP protocols.

Quantum query model. A quantum algorithm V with query access to a bit string $x \in \{0,1\}^n$ is specified by a sequence of unitary operations $V_0 \dots V_q$, that do not depend on the input x. A query to x is given by the unitary U_x on $\log n + 1$ qubits such that

$$U_x |i\rangle |b\rangle = |i\rangle |b \oplus x_i\rangle \ .$$

We denote by V^U the output of V with oracle access to a unitary U. Similarly, $V^U(n)$ denotes the case where V has access to an additional explicit input n.

The final state of an algorithm that makes q queries to the oracle, before measurement, is given by

$$V_q(U_f \otimes I)V_{q-1}(U_f \otimes I) \dots V_1(U_f \otimes I)V_0 |\mathbf{0}\rangle \ .$$

The overall Hilbert space $\mathcal{H}$ used by the algorithm is split into three subspaces $\mathcal{H}_{\mathrm{in}} \otimes \mathcal{H}_{\mathrm{w}} \otimes \mathcal{H}_{\mathrm{out}}$, and we denote by $|\mathbf{0}\rangle = |0\rangle^{\otimes \log(\dim \mathcal{H})}$ the initial state of the verifier's memory. The oracle acts acts on the space $\mathcal{H}_{\mathrm{in}}$, the workspace $\mathcal{H}_{\mathrm{w}}$ can have arbitrary size and $\mathcal{H}_{\mathrm{out}}$ represents the single qubit output of the algorithm. The final step of the algorithm is to measure the $\mathcal{H}_{\mathrm{out}}$ register in the computational basis and return the outcome.

Distance measures. Unless otherwise stated, we will assume the distance measure d for *classical* objects, such as bit strings, as that induced by (normalised) *Hamming weight*: for $x, y \in \{0,1\}^n$, the Hamming weight of x is $|x| \coloneqq |\{i \in [n] : x_i = 1\}|$ and the distance between x and y is $d(x,y) = |\{i \in [n] : x_i \neq y_i\}|$.

For unitary matrices U, V, unless otherwise stated, we consider the distance measure to be that induced by the *Hilbert-Schmidt norm*: the norm of U is $\|U\| \coloneqq \sqrt{\sum_{i=1}^n \sigma_i(U)^2}$, where $\sigma_i(U)$ is the i^{th} eigenvalue of A in nonincreasing order; and the distance between U and V is $d(U,V) = \|U - V\|$.

We say two objects X, Y are *ε-close* if $d(X,Y) \leq \varepsilon$, and otherwise they are *ε-far*. We also say X is ε-close to a set of objects $\{Y_i\}$ if there exists i such that $d(X, Y_i) \leq \varepsilon$.

Property testing. An *interactive proof of proximity* (IPP) for Π is a proof system that solves the approximate decision problem of membership in Π. The *verifier* algorithm receives as input a proximity parameter ε and has oracle access to x. It queries x in at most q coordinates and interacts with an all-powerful but untrusted prover by exchanging m messages, where the total number of communicated bits is c. The verifier must accept when $x \in \Pi$ and reject when x is ϵ-far from Π, with bounded probability of error; the outcome of such an interaction is denoted $\langle P(x), V^x \rangle$. Formally,

Definition 1. *An* interactive proof of proximity *(IPP) for a property* $\Pi = \cup_n \Pi_n$ *is an interactive protocol with two parties: a (computationally unbounded) prover* P *and a verifier* V*, which is a probabilistic algorithm. The parties send messages to each other in turns, with the first sent from prover to verifier, and at the end of the communication, the following two conditions are satisfied:*

1. Completeness: *For every* $\varepsilon > 0$*,* $n \in \mathbb{N}$*, and* $x \in \Pi_n$*, it holds that*

$$\mathbb{P}\left[\langle P(x), V^x \rangle(n, \varepsilon) = 1\right] \geq 2/3 \ ,$$

   where the probability is over the coin tosses of V*.*

2. Soundness: *For every* $\varepsilon > 0$*,* $n \in \mathbb{N}$*,* $x \in \{0,1\}^n$ *that is* ε*-far from* Π_n *and for every computationally unbounded (cheating) prover* P^* *it holds that*

$$\mathbb{P}\left[\langle P^*(x), V^x \rangle(n, \varepsilon) = 1\right] \leq 1/3 \ ,$$

   where the probability is over the coin tosses of V*.*

The query complexity q *of the protocol is the maximum number of queries the verifier makes to* x *in its execution; the* round complexity m *is the number of messages exchanged between prover and verifier; and the* communication complexity c *is the total number of bits communicated by these messages.*

The set of properties Π *for which there exists an IPP protocol with proximity parameter* ε *with* m *messages, communication complexity* c *and query complexity* q *is denoted* $\mathcal{IPP}(\varepsilon, c, q, m)$*.*

When the completeness condition holds with probability 1, i.e., the verifier always accepts when $x \in \Pi$, we call the protocol *one-sided*. Moreover, if the verifier does not receive ε explicitly, but rejects inputs that are ε-far from Π with *detection probability* $\rho(n, \varepsilon) > 0$, the procotol is said to be *proximity-oblivious*.

A *Merlin-Arthur proof of proximity* (MAP) for Π is an IPP where the entire communication is a single message from the prover to the verifier (i.e., an IPP with round complexity 1); the class $\mathcal{MAP}(\varepsilon, p, q)$ is thus defined as $\mathcal{IPP}(\varepsilon, p, q, 1)$. The formal definition of the *quantum* generalisation of MAPs is given in Section 3.1.

A *property tester* is an IPP with $r = 0$, i.e., where no communication occurs. In this case, the verifier is called a *tester*, and we define $\mathcal{PT}(\varepsilon, q) \coloneqq \mathcal{IPP}(\varepsilon, 0, q, 0)$. Moreover

Branching programs. A branching program on n variables is a directed acyclic graph that has a unique source vertex v_0 with in-degree 0 and (possibly) multiple sink vertices with out-degree 0. Each sink vertex is labeled either with 0 (i.e., reject) or 1 (i.e., accept). Each non-sink vertex is labeled by an index $i \in [n]$ and has exactly 2 outgoing edges, which are labeled by 0 and 1. The output of the branching program B on input $x \in \{0,1\}^n$, denoted $B(x)$, is the label of the sink vertex reached by taking a walk, starting at the source vertex v_0, such that at every vertex labeled by $i \in [n]$, the step taken is on the edge labeled by x_i.

A branching program is said to be *read-once* (or ROBP for short) if, along every path from source to sink, every index $i \in [n]$ appears at most once. The size of a branching program B is the number of vertices in its graph.

A branching program is *layered* if its nodes can be partitioned into V_0, V_1, ..., V_ℓ, where V_0 only contains the source node, V_ℓ are the sink nodes and every edge is between V_{i-1} and V_i for some $i \in [\ell]$. The *length* of a layered branching program is its number of (nontrivial) layers ℓ, and its *width* is the maximum size of its layers, i.e., $\max_{i\in[\ell]} \{|V_i|\}$.

Coding theory. 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 k/n and its *distance* is the minimum, over all distinct messages $x, y \in \{0,1\}^k$, of $d(C(x), C(y))$. We shall sometimes slightly abuse notation and use C to denote the set of all of its codewords $\left\{C(x) : x \in \{0,1\}^k\right\} \subset \{0,1\}^n$. If the mapping C is linear (over $\mathbb{F}_2 = \{0,1\}$), we say C is a linear code.

3 QMA Proofs of Proximity

We begin with the formal definition of QMAPs in Section 3.1, and proceed to show, in Section 3.2, a speedup for the general class of *proximity-oblivious* (classical) MAP protocols.

3.1 Definition

A *quantum Merlin-Arthur proof of proximity* (QMAP) for a property $\Pi = \bigcup \Pi_n$ is a proof system consisting of a quantum algorithm V, called a verifier, that is given as explicit input an integer $n \in \mathbb{N}$ and a proximity parameter $\epsilon > 0$. It has oracle access to a unitary $U \in \mathcal{V} \subseteq \mathcal{U}(2^n)$ acting on n qubits, which belongs to a *universe* $\mathcal{V}$ with an associated distance measure.[7] Furthermore, the verifier receives a p-qubit quantum state ρ explicitly as a purported proof that $U \in \mathcal{V}$.

We allow the oracles in the rest of the work to be *quantum oracles*, i.e., CP-Maps. However, using the Stinespring dilation [Sti55, Pau02], we view them as unitary operators again, in a larger Hilbert space, and the formalism generalises readily. This allows us to work with most general transformations allowable by quantum theory.

The verfier V receives n and ε as inputs, and outputs a sequence of unitary operators $V_0 \dots V_q$ that satisfies the two following conditions.

1. *Completeness.* For every $n \in \mathbb{N}$ and $U \in \Pi_n$, there exists a p-qubit quantum state $|\psi\rangle$ such that,[8] for every proximity parameter $\epsilon > 0$,

$$\mathbb{P}\left[V^U(n, \varepsilon, |\psi\rangle) = 1\right] \geq 2/3 \, .$$

Equivalently, *some* p-qubit quantum state $|\psi\rangle$ satisfies

$$\|(|1\rangle\langle 1| \otimes I) W (|\psi\rangle \otimes |\mathbf{0}\rangle)\|^2 \geq 2/3 \, ,$$

[7]Note that this definition generalises classical properties, where $\mathcal{V} = \{U_x : x \in \{0,1\}^n\} \subset \mathcal{U}(2^{n+1})$, the unitary U_x acts as $U_x |i\rangle |b\rangle = |i\rangle |b \oplus x_i\rangle$, and the distance between U_x and U_y is the Hamming distance between x and y.

[8]While the proof can also be a mixed state, assuming it to be pure is without loss of generality; see Remark 1.

where $|\mathbf{0}\rangle$ is the initial state of the verifier and W the unitary obtained by interspersing q calls to the oracle U between the V_i; that is, $W = V_q(U \otimes I)V_{q-1} \dots (U \otimes I)V_0$.

2. *Soundness.* For every $n \in \mathbb{N}$, $\varepsilon > 0$ and p-qubit quantum state $|\psi\rangle$, if $U \in \mathcal{U}(2^n)$ is ε-far from Π_n, then
$$\mathbb{P}\left[V^U(n, \varepsilon, |\psi\rangle) = 1\right] \leq 1/3 \ .$$
Equivalently, *every* p-qubit quantum state $|\psi\rangle$ satisfies
$$\|(|1\rangle\langle 1| \otimes I)W(|\psi\rangle \otimes |\mathbf{0}\rangle)\|^2 \leq 1/3 \ ,$$
where $W = V_q(U \otimes I)V_{q-1} \dots (U \otimes I)V_0$.

The *query complexity* of a QMAP is number of times the verifier calls the oracle U. More precisely, the query complexity is $q = q(n, \varepsilon)$ if, for every $n \in \mathbb{N}$, $\epsilon > 0$ and $U \in \mathcal{U}(2^n)$, the verifier makes q queries to the input. Its *proof complexity* is $p = p(n, \varepsilon)$ if, for every $n \in \mathbb{N}$ and $U \in \Pi_n$, there exists a 2^p-dimensional quantum state $|\psi\rangle$ satisfying both of the above conditions.

The running time $t_V = t_V(n, \varepsilon)$ of the verifier is the minimum depth of the unitaries $V_0 \dots V_q$, composed of gates from a fixed constant-qubit gate set. The running time $t_P = t_P(n, \varepsilon, U)$ of the prover is similarly the minimum depth of the circuit that prepares the proof state $|\psi\rangle$.

Definition 2 ($\mathcal{QMAP}$ complexity class)**.** *Fix a universe set of unitary operators $\mathcal{V}$ and distance measure $d : \mathcal{V} \times \mathcal{V} \to [0, 1]$. $\mathcal{QMAP}(\varepsilon, p, q, t_V, t_P)$ is the class of properties $\Pi \subseteq \mathcal{V}$ that admit a verifier for proximity parameter ε with query complexity q and proof complexity p such that the verifier and prover runtimes are t_V and t_P, respectively.*

The complexity class $\mathcal{QCMAP}(\varepsilon, p, q, t_V, t_P)$ is defined as above, with the additional restriction that the proof be classical*, i.e., that the p-qubit quantum state given as proof is a computational basis state.*

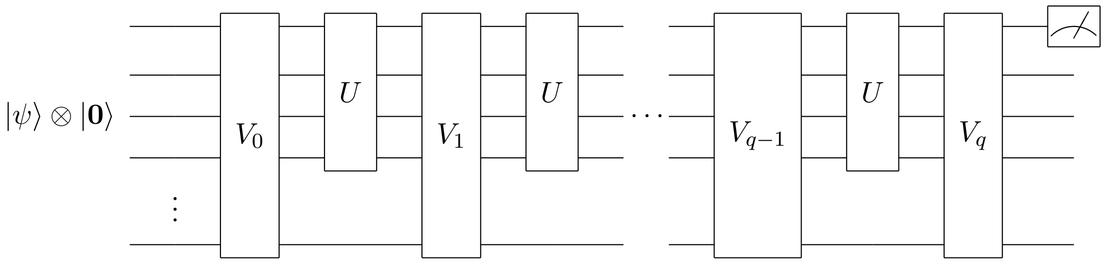

Figure 2: Schematic of a QMAP system that receives as input a proof state $|\psi\rangle$ and makes q queries to a unitary U.

For ease of notation, since the measures of complexity we will use throughout are *proof* and *query* complexity (but *not* time complexity), we will often use $\mathcal{QMAP}(\varepsilon, p, q)$ (and likewise for $\mathcal{QCMAP}$) to denote the class as above, where t_V, t_P are arbitrary functions.

We also denote the class of properties that are ε-testable by quantum testers with q queries as $\mathcal{QPT}(\varepsilon, q)$, that is, $\mathcal{QPT}(\varepsilon, q) := \mathcal{QMAP}(\varepsilon, 0, q)$.

Remark 1 (Proofs are pure states)**.** Without loss of generality, the quantum state given as the proof is a pure state on p qubits, i.e., a rank one positive semi-definite matrix. To see why, note that, if some mixed state $\rho = \sum p_i |\psi_i\rangle \langle\psi_i|$ causes the verifier to accept with probability $2/3$, then, by convexity, there exists a state $|\psi_k\rangle$ in that mixture that would also cause the verifier to output 1 with probability at least $2/3$. Hence, the proof can be the pure state $|\psi_k\rangle$. Likewise, if no pure state can make the verifier accept with probability $2/3$, the same holds for mixed states.

3.2 Quantum speedups for proximity-oblivious MAPs

We begin this section recalling the technique of quantum *amplitude amplification*, and prove its consequences for the classes of algorithms we consider in this work. Roughly speaking, given an algorithm that finds, with probability γ, a preimage of 1 of a boolean function, amplitude amplification allows us to repeat it $O(1/\sqrt{\gamma})$ times in order to find such a preimage high probability (as opposed to $O(1/\gamma)$ repetitions classically). Formally, we have:

Theorem 3.1 ([BHMT02])**.** *Let $v\colon S \to \{0,1\}$ be a Boolean function (from an arbitrary set S) and let A be a quantum algorithm that makes no intermediate measurements (i.e., is a unitary transformation), such that measuring the state $A|\mathbf{0}\rangle$ yields as outcome $s \in v^{-1}(1)$ with probability $\gamma > 0$. Then there exists a quantum algorithm B that uses $O(1/\sqrt{\gamma})$ applications of the unitaries A and A^{-1}, such that measuring $B|\mathbf{0}\rangle$ yields as outcome $s \in v^{-1}(1)$ with probability $2/3$.*

We note that the theorem applies to classical randomised algorithms as a special case. An immediate corollary for *promise problems* in the query model (which is the setting for property testers, MAPs and variations thereof) is the following.[9]

Corollary 4 (Amplitude amplification for promise problems in the query model)**.** *Let $Y, N \subseteq \{0,1\}^n$ with $Y \cap N = \varnothing$ define a promise problem on n-bit strings whose yes- and no-inputs are Y and N, respectively. Let A be a randomised algorithm with oracle access to a string $x \in \{0,1\}^n$ that makes q queries, always accepts when $x \in Y$ and rejects with probability at least γ when $x \in N$. Then, there exists a quantum algorithm B that makes $O(q/\sqrt{\gamma})$ queries to the unitary $U_x|i\rangle|b\rangle = |i\rangle|b \oplus x_i\rangle$, always accepts when $x \in Y$ and rejects with probability $2/3$ when $x \in N$.*

This follows from the observation that each $x \in Y \cup N$ induces a function $f_x\colon \{0,1\}^r \to \{0,1\}$ where r is the number of random bits used by A. If A^x accepts when the outcome of its random coin flips is s, we define $f_x(s) = 0$, and if A^x rejects when its random string is s, then $f_x(s) = 1$. We then apply Theorem 3.1 to the algorithm A^x, for each fixed $x \in Y \cup N$ (or, more precisely, to the modified algorithm that computes f_x written as a reversible circuit and thus implements a query to x as $(i, b) \mapsto (i, b \oplus x_i)$), obtaining B^{U_x} (recall that U_x is the unitary mapping $|i\rangle|b\rangle \mapsto |i\rangle|b \oplus x_i\rangle$). Measuring $B^{U_x}|\mathbf{0}\rangle$, using the outcome as the random string for an execution of A^x and outputting accordingly yields the claimed algorithm.

Note that Corollary 4 directly applies to *one-sided proximity-oblivious* testers, which are testers that always accept n-bit strings in the property and reject strings that are ε-far from it with *detection probability* $\rho(\varepsilon, n)$. We now prove the following observation, which shows that the same speedup can be obtained for MAPs; more precisely, properties that admit one-sided proximity-oblivious MAPs allow for more efficient verification by a quantum algorithm using the same proof string.

[9]While we could state amplitude amplification for testers directly, a subtle issue would arise: MAPs are equivalent to a collection of *partial* testers, which are not “vanilla” testers but are still promise problems.

Theorem 3.2. *Let Π be a property admitting a one-sided proximity-oblivious MAP protocol, which receives a proof of length $p = p(n)$, makes $q = q(n)$ queries and rejects strings ε-far from Π with probability at least $\rho = \rho(\varepsilon, n)$. Then, for any $\varepsilon \in (0, 1)$,*

$$\Pi \in \mathcal{QCMAP}\left(\varepsilon, p, \frac{q}{\sqrt{\rho}}\right) .$$

Proof. Observe that the MAP verifier V can be equivalently described as a collection of probabilistic algorithms $\{V_\pi : \pi \in \{0,1\}^p\}$ indexed by all proof strings π. By definition, for every $x \in \Pi$ there exists $\pi \in \{0,1\}^p$ such that V_π^x always accepts; and, for every x that is ε-far from Π, every proof string π is such that V_π^x rejects with probability at least ρ. Therefore, V_π solves the promise problem whose yes-inputs comprise the subset of Π for which π is a valid proof, and whose no-inputs are the strings ε-far from Π.

Let W_π be the algorithm obtained from V_π by Corollary 4. Then W_π^x accepts (with probability 1) when $x \in \Pi$ and π is a valid proof for x, and W_π^x rejects (with probability $2/3$) when x is ε-far from Π and π is any proof string; in other words, the algorithm W that executes W_π when it receives π as a proof string is a QCMAP verifier for Π. Moreover, since the proof string is reused and W makes $O(q/\sqrt{\rho})$ queries, the proof and query complexities are as stated. □

We conclude with two applications of Theorem 3.2: to *read-once branching programs* (ROBPs) and *context-free languages* (CFLs), which are shown to admit proximity-oblivious MAPs in [GGR18] (see Remark 2 for details on these results).

Theorem 3.3 ([GGR18], Lemma 3.1)**.** *For every read-once branching program on n variables of size $s = s(n)$, let $A_B \coloneqq \{x \in \{0,1\}^n : B(x) = 1\}$ be the set of strings accepted by B. Then, for every $k \leq n$, the property Π_B admits a one-sided proximity-oblivious MAP with communication complexity $O(k \log s)$, query complexity n/k and detection probability $\rho(\varepsilon, n) = \varepsilon$.*

Theorem 3.4 ([GGR18], Lemma 4.5)**.** *For every $k \leq n$, every context-free language L admits a one-sided proximity-oblivious MAP with communication complexity $O(k \log n)$, query complexity n/k and detection probability $\rho(\varepsilon, n) = \varepsilon$.*

Therefore, applying Theorem 3.2 to Theorems 3.3 and 3.4, we obtain:

Corollary 5. *For every read-once branching program B on n variables of size $s = s(n)$, denote by $A_B \coloneqq \{x \in \{0,1\}^n : B(x) = 1\}$ the set of strings accepted by B. Then*

$$A_B \in \mathcal{QCMAP}\left(\varepsilon, O(k \log s), O\left(\frac{n}{k\sqrt{\varepsilon}}\right)\right) \quad \text{for every } k \leq n \text{ and } \varepsilon \in (0,1).$$

Corollary 6. *For every context-free language L,*

$$L \in \mathcal{QCMAP}\left(\varepsilon, O(k \log n), O\left(\frac{n}{k\sqrt{\varepsilon}}\right)\right) \quad \text{for every } k \leq n \text{ and } \varepsilon \in (0,1).$$

Interestingly, these corollaries make explicit a phenomenon in quantum proofs of proximity that does not hold for their classical counterparts: it is possible to test with proximity $\varepsilon = 1/n$, i.e., solve the *exact decision* problem of acceptance by an ROBP or membership in a context-free language, with sublinear proof and query complexity. In particular, taking $k = n^{3/4}$, both complexities are $O(n^{3/4})$. Nonetheless, for the case of branching programs, we will show in Section 4.3 how to lift the read-once restriction and improve on the parameters by directly exploiting *decomposability.*

Remark 2. We note that a context-free language L is defined in terms of an alphabet of *terminals* (which is generally larger than $\{0,1\}$) as well as an alphabet of variables. However, if both alphabets have constant size, we may represent symbols as bit strings with a constant overhead per query; thus Corollary 6 holds for languages over large (constant-size) alphabets.

Moreover, the results of [GGR18] corresponding to Corollaries 5 and 6 are in fact stronger: both apply more generally to IPPs, and thus to MAPs as a special case (see [GGR18, Section 3.2] for details). In addition, the detection probability of the MAP for ROBPs is $\varepsilon n/n'$ if the branching program has an accepting path of length $n' \leq n$; and the MAP for context-free languages works for *partial derivation languages*, a generalisation of CFLs whose strings may include variable symbols as well as terminals.

4 Decomposable properties

In this section, we show how quantum speedups can be applied to proof of proximity protocols for properties that can be broken up into sub-problems in a distance-preserving manner. Roughly speaking, a property Π of n-bit strings is (k,s)-*decomposable* if, using s bits of information (which we call a *specification*), Π can be mapped to k properties $\{\Lambda^{(i)}\}$ and the input string x can be mapped to a set of k strings $\{x^{(i)}\}$ satisfying the following conditions: (1) when $x \in \Pi$, there exists a specification such that $x^{(i)} \in \Lambda^{(i)}$ for all $i \in [k]$; and (2) when x is ε-far from Π, then, for specification, $x^{(i)}$ is roughly ε-far from $\Lambda^{(i)}$ for an average $i \in [k]$.

Definition 3 (Decomposable property)**.** *Let $\Pi = \bigcup \Pi_n$ be a property of bit strings. For $k = k(n)$, $s = s(n)$, $m_1 = m_1(n), \ldots, m_k = m_k(n)$, we say Π is* (k,s)-decomposable *if there exists a mapping from $S \subseteq \{0,1\}^s$ to (possibly distinct) subproperties $\Lambda^{(1)} \subset \{0,1\}^{m_1}, \ldots, \Lambda^{(k)} \subset \{0,1\}^{m_k}$ such that every $x \in \{0,1\}^n$ uniquely determines $x^{(i)} \in \{0,1\}^{m_i}$ satisfying:*[10]

1. *If $x \in \Pi$, then there exists $s \in S$ such that $x^{(i)} \in \Lambda^{(i)}$ for all $i \in [k]$; and*
2. *If x is ε-far from Π, then, for all $s \in S$ and $i \in [k]$, the string $x^{(i)}$ is ε_i-far from $\Lambda^{(i)}$ and $\mathbb{E}_{i \leftarrow \mathcal{D}}[\varepsilon_i] = \Omega(\varepsilon)$, where $\mathcal{D}$ is the distribution over $[k]$ with probability mass $m_i / (\sum_{j \in [k]} m_j)$ on i.*

If $s = O(k \log n)$ we say Π is succinclty *k-decomposable. If the strings $x^{(i)}$ form a partition of x, we say Π is* (k,s)-partitionable.

We remark that succinct k-decompositions are parametrised by $O(k)$ coordinates of the input string. All of our applications are to succincly decomposable properties, and often the $\{x^{(i)}\}$ form an equipartition of x (so each bit of $x^{(i)}$ depends on a single bit of x) and $\mathcal{D}$ is thus the uniform distribution. However, note that if a decomposition is significantly asymmetric, then $\mathcal{D}$ preserves (average) distance while uniform sampling may deteriorate it to $o(\varepsilon)$ (e.g., if $m_i = o(m_1)$ when $i > 1$ and $x^{(1)}$ concentrates all of the corruption).

As we will see in the next sections, decomposable properties enable the construction of efficient proof of proximity protocols and generalise the notion of "parametrised concatenation properties" introduced by [GR18].

[10]We remark that the mappings $\Pi \mapsto (\Lambda^{(1)}, \Lambda^{(2)}, \ldots, \Lambda^{(k)})$ and $x \mapsto (x^{(1)}, x^{(2)}, \ldots, x^{(k)})$ are functions of specification $y \in S$ of the decomposition. Although the notation $x^{(i),y}$ and $\Lambda^{(i),y}$ is formally more accurate, the dependency on y will be clear from context and we omit it for ease of notation.

4.1 Boosting decompositions via amplitude amplification

As the next theorem shows, decomposable properties allow for quantum speedups regardless of whether they admit proximity-oblivious MAPs.

Theorem 4.1. *Let Π be a property that is (k, s)-decomposable into properties of m_i-bit strings, and set $m = \max_{i\in[k]} \{m_i\}$. Suppose each bit of $x^{(i)}$ can be determined by reading b bits of the input string, and each $\Lambda^{(i)}$ admits a one-sided MAP with proximity parameter ε, query complexity $q = q(m, \varepsilon) = m^{\alpha}/\varepsilon^{\beta}$ and proof complexity $p = p(m, \varepsilon)$. Then*

$$\Pi \in \mathcal{QCMAP}(\varepsilon, s + kp, q') ,$$

with

$$q' = \begin{cases} \tilde{O}\left(b \cdot m^{\alpha} \cdot \varepsilon^{-\max(1/2,\beta)}\right) & \text{if } \alpha > 0 \text{ and } \beta \geq 0 \\ \tilde{O}\left(b \cdot m^{1-1/\beta} \cdot \varepsilon^{-1/2}\right) & \text{if } \alpha = 0 \text{ and } \beta \geq 1. \end{cases}$$

Moreover, for exact decision *(i.e., testing with proximity $\varepsilon = 1/n$), a proof of length s and $O(bm\sqrt{k})$ queries suffice.*

Before proceeding to the proof, we note that if the MAP protocols for the subproperties are proximity-oblivious, the query complexity can be improved (see Remark 3). Let us also summarise the proof strategy of [GR18], which we build upon and generalise.

Consider the special case where a property Π is *k-partitionable* and the strings $x^{(i)}$ are simply the substrings of x of length $m = n/k$ which, concatenated, form x. Suppose, moreover, that the subproperties $\Lambda^{(i)}$ admit *testers* with query complexity $q = m^{\alpha}/\varepsilon^{\beta}$ and that Π is the union of $\Lambda^{(1)} \times \Lambda^{(2)} \times \cdots \Lambda^{(k)}$ (over all strings in S). Note that while, in general, a specification must show how to obtain $\Lambda^{(i)}$ from Π *and* how to obtain $x^{(i)}$ from x, for an equipartition the latter is implicit.

A natural candidate for a (classical) MAP protocol for Π is to guess an index $i \in [k]$ and run the tester for $\Lambda^{(i)}$ on $x^{(i)}$. If $x \in \Pi$, then $x^{(i)} \in \Lambda^{(i)}$ for $i \in [k]$ and the tester always accepts; while if x is ε-far from Π, then $x^{(i)}$ is ε_i-far from $\Lambda^{(i)}$ for some ε_i satisfying $\frac{1}{k}\sum_i \varepsilon_i \geq \varepsilon$, *regardless of the specification* (recall that x is ε-far from $\bigcup_{y\in S} \Lambda^{(1)} \times \Lambda^{(2)} \times \cdots \Lambda^{(k)}$).

We now proceed to the proof of the general case, where the decomposition need not be a partition, and it suffices for the subproperties to admit a MAP (rather than a tester). Moreover, we show that quantum algorithms enable a speedup via amplitude amplification (but this requires the MAPs to be *one-sided*, unlike in the classical case).

Proof. Recall that we have a property Π that is (k, s)-decomposable by a collection of strings $S \subseteq \{0,1\}^s$, where each $y \in S$ determines k properties $\Lambda^{(i)} \subseteq \{0,1\}^{m_i}$ and a decomposition of x into k strings $x^{(i)} \in \{0,1\}^{m_i}$. Moreover, each $\Lambda^{(i)}$ admits a MAP with proof complexity p and query complexity $m^{\alpha}/\varepsilon^{\beta}$. The verifier for Π executes the steps shown in Fig. 3.

We note that a more naive strategy would succeed, albeit with a worse dependence on ε: choosing $i \in [k]$ with probability proportional to m_i yields a string $x^{(i)}$ which is $\varepsilon/2$-far from $\Lambda^{(i)}$ with probability at least $\varepsilon/2$, so that one could execute the MAP verifier for $\Lambda^{(i)}$ with proximity parameter $\varepsilon/2$ (and use amplitude amplification to achieve constant soundness by repeating this $O(1/\sqrt{\varepsilon})$ times). However, the technique of *precision sampling* [Lev85] overcomes the issue of not knowing the distances ε_i between $x^{(i)}$ and $\Lambda^{(i)}$ more economically: trying every proximity parameter 2^j with $j \in [O(\log 1/\varepsilon)]$, in the spirit of binary search, incurs a merely logarithmic overhead.

Input: explicit access to a proximity parameter $\varepsilon > 0$ and a proof string $\pi \in \{0,1\}^{s+kp}$, and oracle access to a string $x \in \{0,1\}^n$.

1. Interpret the proof as a concatenation of a string $y \in \{0,1\}^s$ with k strings $\pi_1, \dots, \pi_k \in \{0,1\}^p$. If $y \notin S$, i.e., y does not specify a decomposition, then reject.
2. For every $j \in [\lceil \log 1/\varepsilon \rceil + 1]$, let M_j be the algorithm obtained from Corollary 4 by performing $O\left(\sqrt{\frac{\log 1/\varepsilon}{2^j \varepsilon}}\right)$ rounds of amplitude amplification to the following subroutine:
   (a) Sample $i \in [k]$ with probability $m_i / \sum_{j \in [k]} m_j$ and run the MAP verifier for $\Lambda^{(i)}$ on input $x^{(i)}$, with proximity parameter 2^{-j}, using π_i as the proof string. Reject if the MAP for $\Lambda^{(i)}$ rejects.
3. Execute M_j for every $j \in [O(\log 1/\varepsilon)]$. If any of them rejects, then reject; otherwise, accept.

Figure 3: MAP verifier for a (k,s)-decomposable property Π

Completeness follows immediately: if $x \in \Pi$, then there exists a string $y \in S$ such that x determines $x^{(i)} \in \Lambda^{(i)}$ for all $i \in [k]$. Since the properties $\Lambda^{(i)}$ admit one-sided MAP protocols with proof complexity p, the procedure in Item 2(a) always accepts when given the proof string $\pi = (y, \pi_1, \dots, \pi_k)$, where π_i is a valid proof for $x^{(i)}$. Therefore, the verifier always accepts as well.

Now, suppose x is ε-far from Π and the proof string $\pi = (y, \pi_1, \dots, \pi_n)$ is such that $y \in S$ (since otherwise the verifier rejects immediately). Then, since Π is decomposable, $x^{(i)}$ is ε_i-far from $\Lambda^{(i)}$ and $\mathbb{E}_{i \leftarrow \mathcal{D}}[\varepsilon_i] = \Omega(\varepsilon)$, where $\Lambda^{(i)}$ are the subproperties defined by y and $\mathcal{D}$ is the distribution over $[k]$ that samples i with probability proportional to m_i.

To show soundness, we will make use of the following (precision sampling) lemma.

Lemma 1 ([Gol14, Fact A.1])**.** *There exists $j \in \left[\left\lceil \log \frac{1}{\varepsilon} \right\rceil + 1\right]$ such that*

$$\mathbb{P}_{i \leftarrow \mathcal{D}}[\varepsilon_i \geq 2^{-j}] = \Omega\left(\frac{2^j \varepsilon}{\log 1/\varepsilon}\right) .$$

If the procedure in Item 2(a) samples $i \in [k]$ such that $\varepsilon_i \geq 2^{-j}$, then it rejects with probability 2/3 (since the MAP has soundness 2/3). With j as ensured by Lemma 1, the probability it samples such an $i \in [k]$ is $\Omega\left(\frac{2^j \varepsilon}{\log 1/\varepsilon}\right)$, so that the probability it rejects is $\frac{2}{3} \cdot \Omega\left(\frac{2^j \varepsilon}{\log 1/\varepsilon}\right) = \Omega\left(\frac{2^j \varepsilon}{\log 1/\varepsilon}\right)$; therefore, the algorithm M_j obtained from Corollary 4 by $O\left(\sqrt{\frac{\log 1/\varepsilon}{2^j \varepsilon}}\right)$ rounds of amplitude amplification rejects, causing the verifier to also reject, with probability 2/3.

We now prove the stated upper bounds on the query complexity. For every j, each execution of the MAP verifier for $\Lambda^{(i)}$ makes $q(m, 2^{-j})$ queries to $x^{(i)}$, which translate into $b \cdot q(m, 2^{-j})$ queries to x (since each query to $x^{(i)}$ can be emulated with b queries to x). The total query complexity is therefore

$$\sum_{j \in [\lceil \log 1/\varepsilon \rceil + 1]} \sqrt{\frac{\log 1/\varepsilon}{2^j \varepsilon}} \cdot b \cdot q\left(m, 2^{-j}\right) = \tilde{O}\left(\frac{b}{\sqrt{\varepsilon}} \sum_{j \in [\lceil \log 1/\varepsilon \rceil + 1]} \frac{q\left(m, 2^{-j}\right)}{2^{j/2}}\right) .$$

If $q(m, \varepsilon) = m^\alpha / \varepsilon^\beta$ with $\alpha > 0$ and $\beta \geq 0$, then

$$\tilde{O}\left(\frac{b}{\sqrt{\varepsilon}}\sum_{j\in[\lceil\log 1/\varepsilon\rceil+1]}\frac{q\left(m,2^{-j}\right)}{2^{j/2}}\right)=\tilde{O}\left(\frac{bm^{\alpha}}{\sqrt{\varepsilon}}\sum_{j\in[\lceil\log(1/\varepsilon)\rceil+1]}2^{j\left(\beta-\frac{1}{2}\right)}\right)$$
$$=\tilde{O}\left(bm^{\alpha}\varepsilon^{-\max(1/2,\beta)}\right)\ .$$

If $\alpha=0$ and $\beta>0$, we use the bound $q(m,2^{-j})\leq m$ for all ε (from the trivial tester that queries the entire input), and the query complexity becomes

$$\tilde{O}\left(\frac{b}{\sqrt{\varepsilon}}\sum_{j\in[\lceil\log 2/\varepsilon\rceil]}\frac{q\left(m,2^{-j}\right)}{2^{j/2}}\right)=\tilde{O}\left(\frac{b}{\sqrt{\varepsilon}}\sum_{j\in[\lceil\log 2/\varepsilon\rceil]}\min\left\{\frac{m}{2^{j/2}},2^{j\left(\beta-\frac{1}{2}\right)}\right\}\right)$$
$$=\tilde{O}\left(\frac{b}{\sqrt{\varepsilon}}\sum_{j\in[\lceil\log 2/\varepsilon\rceil]}m^{1-1/\beta}\right)$$
$$=\tilde{O}\left(\frac{bm^{1-1/\beta}}{\sqrt{\varepsilon}}\right)\ .$$

Finally, observe that, for testing with $\varepsilon=1/n$ (i.e., deciding exactly), one may take the MAPs for $\Lambda^{(i)}$ to be the trivial testers (with query complexity m and no proof). Moreover, it is unnecessary to iterate over j and apply Lemma 1; sampling $i\in[k]$ uniformly and running the trivial tester requires bm queries to x leads to a rejection with probability at least $1/k$, since $\frac{1}{k}\sum_{i\in[k]}\varepsilon_i>0$ implies $x^{(i)}\notin\Lambda^{(i)}$ for at least one $i\in[k]$. Therefore, applying $O(\sqrt{k})$ rounds of amplitude amplification to this procedure ensures rejection of $x\notin\Pi$ with constant probability and yields query complexity $O(bm\sqrt{k})$. □

Remark 3 (Additional speedup for proximity-oblivious MAPs)**.** If the MAP verifiers for the subproperties $\Lambda^{(i)}$ are proximity-oblivious, it is possible to improve on the query complexity of Theorem 4.1 significantly. More precisely, suppose each $\Lambda^{(i)}$ admits a proximity-oblivious MAP with query complexity $O(1)$ and detection probability $\rho(\varepsilon,m)=\varepsilon^{\beta}/m^{\alpha}$. Then, if an input is ε-far from Π, for some $j\in[O(\log 1/\varepsilon)]$ (as ensured by Lemma 1), the procedure of Item 2(a) samples $i\in[k]$ such that $x^{(i)}$ is ε_i-far from $\Lambda^{(i)}$ with $\varepsilon_i=\tilde{\Omega}(2^j\varepsilon)$.

The procedure thus rejects in this case with probability $\tilde{\Omega}(2^j\varepsilon)\cdot\rho(2^{-j},m)=\tilde{\Omega}(2^{j(1-\beta)}\varepsilon/m^{\alpha})$. By applying $\tilde{O}(m^{\alpha/2}/\sqrt{2^{-j(\beta-1)}\varepsilon})$ rounds of amplitude amplification for each j (instead of $\tilde{O}(1/\sqrt{2^j\varepsilon})$ performed for MAPs that are not proximity-oblivious), and, since $2^{-j}=\Omega(\varepsilon)$, the total query complexity becomes $\tilde{O}(b\sqrt{m^{\alpha}/\varepsilon^{\beta}})$.

We finish this section with a corollary of Theorem 4.1 in the *graph orientation model*. A directed graph is called *Eulerian* if the in-degree of each of its vertices is equal to its out-degree. An orientation is a mapping from the edges to $\{0,1\}$, representing whether each edge is oriented from i to j or from j to i, and the distance between two orientations is the fraction of edges whose orientation must be changed to transform one into the other. Let Π_E be the property consisting of all Eulerian orientations of the complete bipartite graph $K_{2,n-2}$, i.e., the graph with vertex set $[n]$ and edge set $\{\{i,j\}:i\leq 2,j\geq 3\}$.

Corollary 7. *The property Π_E has a one-sided QCMAP, with respect to proximity parameter ε, that uses a proof of length $O(k \cdot \log n)$ and has query complexity $\tilde{O}\left(\frac{n}{k\sqrt{\varepsilon}}\right)$.*

We remark that [GR18] obtain a MAP protocol with the same proof complexity and query complexity $\tilde{O}(n/(\varepsilon k))$, by applying their (classical) version of Theorem 4.1 using the trivial tester for each of the subproperties. However, the query complexity of the classical MAP becomes linear with $\varepsilon = \Omega(1/k)$, whereas the QCMAP is able to decide *exactly* (i.e. test with $\varepsilon = 1/n$) with query complexity $O(n^{3/2}/k)$ (which is still sublinear whenever $k = \omega(\sqrt{n})$). In particular, with $k = n^{3/4}$, both query and communication complexities are $\tilde{O}(n^{3/4})$; see Section 4.3 for further discussion and applications of Theorem 4.1 to exact decision problems.

4.2 k-monotonicity

In this section, we show that a generalisation of monotonicity of Boolean functions over the line $[n]$ is efficiently testable by QCMAP protocols. A function $f : [n] \to \{0,1\}$ is *k-monotone* if any sequence of integers $1 \leq x_1 < x_2 \ldots < x_\ell \leq n$ such that $f(x_1) = 1$ and $f(x_i) \neq f(x_{i+1})$ for all $i < \ell$ has length $\ell \leq k$. This problem was studied in [CGG+19], where one-sided ε-testers for k-monotonicity on the line are shown to require $\Omega(k/\varepsilon)$ queries, while two-sided testers can achieve query complexity $\tilde{O}(1/\varepsilon^7)$ (which, although a far worse dependence on ε than the lower bound, is *independent* of k).

In order to apply Theorem 4.1, we must only show that k-monotone functions are decomposable. Define $\Pi_{k,[n]}$ as the set of k-monotone Boolean functions on the line $[n]$.[11] Then,

Theorem 4.2. *For any $k \in [n]$, the property $\Pi_{k,[n]}$ is k-decomposable.*

Proof. Since a k-monotone function f has at most $k-1$ critical points, where it changes from nondecreasing to nonincreasing or vice-versa, specifying these points yields a decomposition of f into (1-)monotone subfunctions.

More precisely, a string of length $s \leq (k-1)\log n$ determines a decomposition of the input f by specifying $\ell \leq k-1$ integers $1 < n_1 < n_2 < \cdots < n_\ell < n$. Define $n_0 = 1$, $n_{\ell+1} = n$, $m_i := n_i - n_{i-1} + 1$ and the function $f_i : [m_i] \to \{0,1\}$ by $f_i(x) = f(x + n_{i-1} - 1)$ for all $i \in [\ell+1]$. Then, $f \in \Pi_{k,[n]}$ if and only if:

1. for $i \in [\ell+1]$ odd, $f_i \in \Lambda^{(i)} := \{g \in \Pi_{1,m_i} : g(1) = f_i(1)\}$; and
2. for $i \in [\ell+1]$ even, $f_i \in \Lambda^{(i)} := \{1 - g : g \in \Pi_{1,m_i} \text{ and } g(1) = 1 - f_i(1)\}$.

It is clear that, when $f \in \Pi_{k,[n]}$, there exists a set of $\ell \leq k-1$ distinct integers in $[2, n-1]$ that satisfies both conditions. When f is ε-far from $\Pi_{k,[n]}$, the sum of absolute distances $\varepsilon_i m_i$ from each f_i to $\Lambda^{(i)}$ is $\sum_i \varepsilon_i m_i \geq \varepsilon n$. Therefore,

$$\mathbb{E}_{i \leftarrow \mathcal{D}}[\varepsilon_i] \geq \varepsilon \cdot \frac{n}{\sum_{i\in[\ell+1]} m_i} = \varepsilon \cdot \frac{n}{n+\ell} = \Omega(\varepsilon) \ ,$$

where $\mathcal{D}$ is the distribution over $[\ell+1]$ that has probability mass $m_i/(\sum_{j\in[\ell+1]} m_j)$ at point i.

[11] We consider the standard representation of a Boolean function as the bit string obtained by concatenating all function evaluations, i.e., $f : [n] \to \{0,1\}$ is represented by $x \in \{0,1\}^n$ with $x_i = f(i)$.

Finally, while this ensures $(\ell+1)$-decomposability for some $\ell \leq k-1$, one may deterministically transform it into a k-decomposition (and, in fact, a K-decomposition for any $K \geq \ell$) by, for example, iteratively finding the largest interval and dividing it at its midpoint $k-\ell-1$ times (with the requirement that nonincreasing functions in the large interval are monotone nonincreasing in both subintervals, and likewise for the nondecreasing case). $\square$

Since monotonicity on the line $[m]$ is ε-testable with $q(m,\varepsilon) = O(1/\varepsilon)$ queries [Gol17, Proposition 1.5], applying (the second case of) Theorem 4.1 yields Corollary 1: k-monotonicity has a QCMAP protocol with proof complexity $O(k \log n)$ and query complexity $\tilde{O}(1/\sqrt{\varepsilon})$.

It is worth noting that the standard monotonicity tester on the line is *not* proximity-oblivious, unlike, e.g., on the Boolean hypercube, where both the "edge tester" [GGL+00] and the state-of-the-art [KMS18] are proximity-oblivious; thus, one could not directly apply amplitude amplification, and must exploit decomposability via Theorem 4.1.

Moreover, the picture changes when comparing one-sided QCMAPs with *two-sided* testers and MAPs: the two-sided k-monotonicity tester of [CGG+19] has query complexity $\tilde{O}(1/\varepsilon^7)$, which is independent of k. However, the QCMAP outperforms this tester even with mild dependencies of the proximity parameter ε on n. Indeed, when $\varepsilon = o(1/\log^{1/7} n)$ and $k = O(1)$, the tester's query complexity is superlogarithmic while the proof and query complexities of the QCMAP are logarithmic; in particular, the QCMAP yields an exponential improvement when $\varepsilon = O(1/n^{\gamma})$ for any $\gamma \in (0,1]$.

With the transformation from two-sided testers to one-sided MAPs of [GR18], it is also possible to compare our one-sided QCMAP against one-sided MAPs. From the aforementioned two-sided tester, one obtains a MAP with proof complexity $\operatorname{polylog} n$ and query complexity $O(\operatorname{polylog}(n/\varepsilon)/\varepsilon^7)$. Thus, the QCMAP continues to offers strong advantages except when k and ε are large (e.g., $k = \omega(\operatorname{polylog} n)$ and $\varepsilon = \Omega(1)$).

4.3 Exact problems

To conclude the discussion of decomposability and its consequences, we shift focus to a special case: that of testing n-bit strings with proximity parameter $\varepsilon = 1/n$. Since, for any $\Pi \subseteq \{0,1\}^n$ and $x \in \{0,1\}^n \setminus \Pi$, the string x is at least $1/n$-far from Π, this is the task of *exactly* deciding membership in Π.

Observe that for classical MAPs, nontrivial properties require $\Omega(n)$ queries in this case: even if a verifier receives as proof a claim x' that is allegedly equal to its input string, it requires $O(1/\varepsilon) = \Omega(n)$ queries to check the validity of the claim. Remarkably, *quantum* algorithms are able to solve exact decision problems with sublinear queries (as illustrated by Grover's algorithm, which makes $O(\sqrt{n})$ queries). Thus, as we show next, insights arising from decomposability are applicable to this setting. We begin by showing a nontrivial QCMA protocol for the parity of a bit string in Section 4.3.1, and then extend it to *branching programs* in Section 4.3.2.

4.3.1 Parity

Consider the problem of deciding if an n-bit string has even parity. This is clearly maximally hard, requiring $\Omega(n)$ queries even for *interactive proofs with arbitrary communication*, which we show next for completeness.

Lemma 2. *Any IP verifier that accepts strings of even parity and rejects strings of odd parity with probability* $2/3$ *must make at least* $n/3$ *queries to its input.*

Proof. Let V and P be a verifier and an honest prover for an IP for parity, and assume, towards contradiction, that the query complexity of V is less than $n/3$.

Fix an arbitrary input $x \in \{0,1\}^n$ of even parity. Define S_x as the random variable comprising all the coordinates queried by V in an execution $\langle V^x, P(x)\rangle$ of the protocol, and let $I \in [n]$ be a uniform random variable independent from S_x. Then, since $|S_x| < n/3$,

$$\frac{1}{n}\sum_{i=1}^{n} \mathbb{P}[i \in S_x] = \sum_{i=1}^{n} \mathbb{P}[I = i] \cdot \mathbb{P}[I \in S_x \mid I = i] = \mathbb{P}[I \in S_x] < \frac{1}{3} ,$$

so there exists $i \in [n]$ such that

$$\mathbb{P}[V^x \text{ queries } i \text{ in the execution } \langle V^x, P(x)\rangle] = \mathbb{P}[i \in S_x] < \frac{1}{3} .$$

Now, consider the execution of a protocol on input $y \in \{0,1\}^n$ obtained by flipping the i^{th} bit of x (i.e., such that $y_j = x_j$ if $j \neq i$ and $y_i = 1 - x_i$ otherwise). Let $\tilde{P}$ be a (malicious) prover that executes on y exactly as P does on x; that is, set $\tilde{P}(y) = P(x)$. We thus have

$$\begin{aligned}\mathbb{P}[\langle V^y, \tilde{P}(y)\rangle \text{ rejects}] = \mathbb{P}[i \in S_y] \cdot \mathbb{P}[\langle V^y, \tilde{P}(y)\rangle \text{ rejects} \mid i \in S_y] \\ + \mathbb{P}[i \notin S_y] \cdot \mathbb{P}[\langle V^y, \tilde{P}(y)\rangle \text{ rejects} \mid i \notin S_y] ,\end{aligned}$$

and, moreover, the following equalities between events hold:

$$\begin{aligned}[i \notin S_y] = [i \notin S_x] \text{ , and thus} \\ [\langle V^y, \tilde{P}(y)\rangle \text{ rejects} \mid i \notin S_y] = [\langle V^x, P(x)\rangle \text{ rejects} \mid i \notin S_x] .\end{aligned}$$

Therefore,

$$\begin{aligned}\mathbb{P}[\langle V^y, \tilde{P}(y)\rangle \text{ rejects}] &< \frac{1}{3} + \mathbb{P}[i \notin S_y] \cdot \mathbb{P}[\langle V^y, \tilde{P}(y)\rangle \text{ rejects} \mid i \notin S_y] \\ &= \frac{1}{3} + \mathbb{P}[\langle V^x, P(x)\rangle \text{ rejects and } i \notin S_x] \\ &\leq \frac{1}{3} + \mathbb{P}[\langle V^x, P(x)\rangle \text{ rejects}] \leq \frac{2}{3} ,\end{aligned}$$

contradicting the correctness of the protocol. □

Rather surprisingly, however, there exists a quantum *non-interactive* protocol that exploits amplitude amplification and achieves sublinear query and communication complexities. This is a direct consequence of the following.

Proposition 1. *For any* $k \leq n$*, the property* $\Pi := \left\{x \in \{0,1\}^n : \bigoplus_{j\in[n]} x_j = 0\right\}$ *is succinctly* k*-partitionable.*

Proof. The set of strings that specify decompositions is $S = \left\{ y \in \{0,1\}^k : \bigoplus_{i\in[k]} y_i = 0 \right\}$, i.e., the set of k-bit strings of even parity. A string $y \in S$ specifies a decomposition where, for each $i \in [k]$, the i^{th} subproperty is $\Lambda^{(i)} = \left\{ x^{(i)} \in \{0,1\}^{n/k} : \bigoplus_{j\in[n/k]} x^{(i)}_j = y_i \right\}$ and $x \in \{0,1\}^n$ induces $x^{(i)}$ as the substring of x consisting of the i^{th} block of n/k bits, i.e., $x^{(i)} = \left(x_{\frac{(i-1)n}{k}+1}, x_{\frac{(i-1)n}{k}+2}, \dots, x_{\frac{in}{k}} \right)$.

Note that the condition $x^{(i)} \in \Lambda^{(i)}$ for all $i \in [k]$ uniquely defines $y \in \{0,1\}^k$ by $y_i = \bigoplus_{\frac{(i-1)n}{k} < j \le \frac{in}{k}} x_j$. Therefore, if x has even parity, there exists $y \in S$ satisfying the condition, while if $x \notin \Pi$ the only string that satisfies it is not in S. Thus x is $1/n$-far from Π and, *for all* $y \in S$, there exists $i \in [k]$ such that $x^{(i)} \notin \Lambda^{(i)}$, so that the distances ε_j from $x^{(j)}$ to $\Lambda^{(j)}$ satisfy $\frac{1}{k}\sum_{j\in[k]} \varepsilon_j \ge \frac{\varepsilon_i}{k} \ge \frac{1}{k\cdot\frac{n}{k}} = 1/n$. $\square$

Applying Theorem 4.1 with $k = n^{2/3}$, we have:

Corollary 8. *There exists a QCMA protocol for parity with $O(n^{2/3})$ query and communication complexities.*

4.3.2 Branching programs

Recall that Corollary 5 shows membership in the set of strings accepted by *read-once* branching programs can be decided with $O(n^{3/4})$ query and proof complexities; thus, since parity is computed by an ROBP (of width 2), we obtain a QCMA protocol with the same parameters. Observe, however, that this is significantly worse than the $O(n^{2/3})$ upper bound of the previous section, which directly exploits decomposability. This suggests a similar improvement may be possible for branching programs, which we now show to be indeed the case for *layered* branching programs parametrised by their *length* ℓ and *width* w (see Section 2 for the definitions).

Proposition 2. *Let B be a layered branching program on n-bit strings of length $\ell = \ell(n)$ and width $w = w(n)$. For any $k \le \ell$, the set $A_B \subseteq \{0,1\}^n$ of strings accepted by B is $(k, O(k \log w))$-partitionable.*

Proof. Let V be the vertex set of the graph that defines the branching program B, whose layers are $V_0 = \{v_0\}$, V_1, $\dots$, V_ℓ and whose set of accepting nodes is $F \subseteq V_\ell$. Decompositions are specified by the set (of $k \log w$-bit representations of) $S = \left\{ (v_1, \dots, v_k) : v_i \in V_{i\ell/k} \text{ and } v_k \in F \right\}$. (Note that $|V_i| \le w$ implies $\log w$ bits suffice to identify each vertex in a given layer.) A specification fixes k nodes that partition an alleged accepting path into equal parts of length ℓ/k. Thus, $\Lambda^{(i)}$ is the set of strings accepted by the branching program that is the subgraph of B with layers $\{v_{i-1}\}$ and V_j for $\frac{(i-1)\ell}{k} < j \le \frac{i\ell}{k}$, source node v_{i-1} and accepting node v_i. The string $x^{(i)}$ is the i^{th} block of ℓ/k bits of the input x.

If $x \in A_B$, there clearly exists an accepting path $(v_0, u_1, \dots, u_\ell)$, namely the path determined by the execution of B on x. Therefore, the specification $y = (u_{\ell/k}, u_{2\ell/k}, \dots, u_{(k-1)\ell/k}, u_\ell) \in S$ satisfies $x^{(i)} \in \Lambda^{(i)}$ for all $i \in [k]$. If $x \notin A_B$, on the other hand, every specification is such that at least one sub-path does not match the corresponding sub-path determined by x (since otherwise x would be accepted by B), implying $x^{(i)} \notin \Lambda^{(i)}$ for some i. $\square$

Applying Theorem 4.1 with $k = \ell^{2/3}$, we obtain Theorem 3: width-w, length-ℓ branching programs admit QCMAP protocols with proof complexity $O(\ell^{2/3} \log w)$ and query complexity

$O(\ell/\sqrt{k}) = O(\ell^{2/3})$. Note that the $\Omega(n)$ classical complexity lower bound for parity implies the same bound for width-2 branching programs of length n, and that the complexities of the QCMA protocol are sublinear up to length $o(n^{3/2}/\log w)$; in particular, this holds for width-2^{n^α} branching programs of length $O(n^{3/2-\alpha})$ for every $\alpha \geq 0$. Thus, besides lifting the read-once restriction, it improves on Corollary 5 for a wide range of parameters.

5 Bipartiteness in bounded-degree graphs

In the bounded-degree graph model, an algorithm is given query access to the *adjacency list* of a graph G whose vertices have their degree bounded by $d = O(1)$. More precisely, given a vertex v and an index $i \in [d]$, the oracle corresponds to the mapping $(v, i) \mapsto w$, where w is the i^{th} neighbour of v (if it exists) or $w = \perp$ (if v has fewer than i neighbours). The distance between two graphs is the fraction of pairs (i, v) whose outputs differ between the two adjacency list mappings.

Our goal in this section will be to construct a QMAP protocol for testing whether a bounded-degree *rapidly-mixing* graph G (i.e., where the last vertex in a random walk of sufficiently large length $\ell = O(\log n)$ starting from any vertex is distributed roughly uniformly), given as an adjacency list oracle, is bipartite or ε-far from every (rapidly-mixing) bipartite graph. Note that this differs from the standard testing setting on graphs by the additional restriction that *both* yes- and no-inputs be rapidly mixing: whether G is bipartite or ε-far from bipartite, for any pair u, v of vertices, the random walk starting from u ends at v with probability between $1/(2n)$ and $2/n$.

Our QMAP protocol for bipartiteness builds on the MAP protocol of [GR18, Theorem 7.1], modifying it to make it amenable to a quantum speedup. In the strategy laid out in that work, the classical verifier receives as proof a set S of k vertices that are allegedly on the same side of a bipartition. It repeatedly samples a uniformly random vertex, takes many lazy random walks of length ℓ and,[12] for each of them, records if it stops at a vertex in S as well as the parity of the number of non-lazy steps (where the walk moves to another vertex). If two walks starting from the same vertex end in S with different parities, the verifier has found a witness to the fact that the graph is not bipartite and rejects.

For any proximity parameter ε and $k \leq n/2$, the MAP protocol of [GR18] requires a proof of length $k \log n$ and makes $O(n\varepsilon^{-2}/k)$ queries. By making the verifier quantum (but using the same classical proof), we obtain improvements in the dependence on both n/k and ε:

Theorem 5.1. *For every $k \leq n/2$, there exists a one-sided QCMAP for deciding whether an n-vertex bounded-degree graph G is bipartite or ε-far from being bipartite, under the promise that G is rapidly-mixing, with proof complexity $k \cdot \log n$ and query complexity $\tilde{O}((n/k)^{2/3} \cdot \varepsilon^{-4/3})$.*

Our strategy to obtain a quantum speedup uses the quantum *collision-finding* algorithm [Amb07, ACL11] to a suitable modification of the verifier strategy outlined above.[13] We note that the function f to which we will apply the collision-finding algorithm is *not* the adjacency list oracle, but one whose unitary representation can be obtained by adjacency list queries and classical computation.

[12] A lazy random walk moves from a vertex v to a uniform random neighbour with probability $\frac{d_v}{2d}$ and otherwise stays at v, where d_v is the degree of v and $d = O(1)$ is the graph's degree bound.

[13] We remark that collision-finding here refers to a generalisation of the *element distinctness* algorithm to symmetric relations beyond equality. In particular, it is *not* a quantum algorithm for the "r-to-1 collision problem", where (many) collisions are promised to exist.

Theorem 5.2 (Quantum collision-finding [ACL11, Theorem 9])**.** *Let $f\colon X \to Y$ be a function given via a unitary oracle and $R \subseteq Y \times Y$ be a symmetric binary relation. There exists a quantum algorithm that makes $O(|X|^{2/3} \operatorname{polylog} |Y|)$ queries to f, always accepts if $(x, x') \notin R$ for every distinct $x, x' \in X$, and rejects with probability $8/9$ if there exist distinct $x, x' \in X$ such that $(x, x') \in R$.*

Returning to the aforementioned classical MAP, observe that the information of each set of t random walks of length ℓ starting from a fixed vertex can be represented by a function $f\colon T \to \{0,1\}^2$, where $T \subset \{0,1\}^{\ell'}$ is composed of t strings of length $\ell' = O(\ell)$ (note that each step of the random walk can be performed with $O(1)$ bits, for a total of $\ell' = O(\ell)$ random coins per walk) and $f(r)$ encodes whether the walk reaches a vertex in S as well as its parity when its inner randomness corresponds to the string r. While applying the collision-finding algorithm to this function *for each fixed vertex* already yields a speedup [ACL11], it can be improved by first sampling all the starting vertices for the random walks and augmenting the domain of f with this set, as we show next.

Proof of Theorem 5.1. We take the classical MAP outlined earlier and apply collision-finding to a function f whose domain includes *all* of the random choices of the classical algorithm: the domain is $W \times T$, where W is a (random) subset of starting vertices from which we take random walks determined by the (random) set T of sequences of coin flips. The proof specifies a subset $S \subseteq L$ of size k, for some L (allegedly) defining a bipartition $V = L \cup R$ of G with $|L| \geq |R|$. The resulting QCMAP verifier is shown in Fig. 4.

Input: oracle access to the adjacency list of an n-vertex graph G, as well as explicit access to a proximity parameter $\varepsilon > 0$ and a proof string $\pi \in \{0,1\}^{k \log n}$ (representing a set $S \subset V$ of size k).

1. Select a subset $W \subset V$ by sampling $O(\varepsilon^{-1} \log n)$ vertices uniformly and independently.
2. Select a subset $T \subset \{0,1\}^{\ell'}$, where $\ell' = O(\ell)$ and $\ell = O(\log n)$, by sampling $O\left(\frac{n}{k} \cdot \frac{\log n}{\varepsilon}\right)$ bit strings uniformly and independently.
3. Let $f : W \times T \to W \times \{0,1\}^2$ be the function computed by the following subroutine:
    (a) Let (w, r) with $w \in W$ and $r \in T$ be the input. Take the (lazy) random walk of length ℓ starting at w using r as the random bits. Let $a \in \{0,1\}$ be the indicator of whether the walk stops at a vertex in S and $b \in \{0,1\}$ be the parity of the walk (i.e., the number of non-lazy steps). Return (w, a, b).
4. Execute the algorithm of Theorem 5.2 with respect to f to find (v, t) and (v', t') such that $f(v, t) = (v, a, b)$ and $f(v', t') = (v', a', b')$ with $v = v'$, $a = a' = 1$ and $b \neq b'$. Reject if such a pair is found, and accept otherwise.

Figure 4: QCMAP verifier for bipartiteness of bounded-degree graphs

First, note that the function f contains a collision with respect to the relation $R \subset (W \times \{0,1\}^2)^2$ that comprises all pairs of triples of the type $((w,1,b),(w,1,1-b))$ *if and only if* there are two random walks of length ℓ that end at a vertex in S with different parities.

If G is bipartite with vertex set $L \cup R$ and $S \subseteq L$, every path with both endpoints in S has even length. But the existence of two paths from the same vertex into S with different parities implies the existence of a path of odd length with both endpoints in S; therefore, since the function f does not contain a collision, the verifier always accepts when G is bipartite and the proof consists of k elements on the same side of a bipartition.

If G is ε-far from bipartite, then [GR18] (building on [GR99]) show the following: defining a as the indicator of whether a random walk of length $\ell = O(\log n)$ starting from $v \in V$ stops at a vertex in S, and b as the parity of this random walk, then a fraction of $\Omega(\varepsilon/\log n)$ vertices $w \in V$ are such that *both* $\mathbb{P}[a = 1, b = 0 \mid v = w]$ and $\mathbb{P}[a = 1, b = 1 \mid v = w]$ are $\Omega(\frac{k\varepsilon}{n \log n})$. For a sufficiently large number of starting vertices $|W| = O(\frac{\log n}{\varepsilon})$, with probability 8/9 there exists a vertex $w \in W$ satisfying this condition. If such w was sampled, in a sufficiently large number $|T| = O(\frac{n \log n}{k\varepsilon})$ of random walks starting from each vertex, there exists a pair of length-ℓ random walks that start from w and end in S with different parities with probability 8/9. Finally, since the algorithm of Theorem 5.2 rejects with probability 8/9 if the function f contains a collision, the verifier rejects except with probability $3 \cdot 1/9 = 1/3$ (by a union bound over the complements of the three events) and soundness follows.

The quantum collision-finding algorithm computes $O((|W| \cdot |T|)^{2/3})$ times the function f, each of which simulates a random walk of length $\ell = O(\log n)$. Each step of the random walk requires $O(1)$ queries to the graph G, so that $O(\log n)$ queries are made per walk. The query complexity of the verifier is therefore

$$\begin{aligned} O\left((|W| \cdot |T|)^{2/3} \cdot \ell \cdot \operatorname{polylog} |W|\right) &= \tilde{O}\left(\left(\frac{n}{k\varepsilon} \cdot \frac{1}{\varepsilon}\right)^{2/3}\right) \\ &= \tilde{O}\left(\left(\frac{n}{k}\right)^{2/3} \cdot \frac{1}{\varepsilon^{4/3}}\right) . \end{aligned}$$ □

We remark that classically testing bipartiteness in the bounded-degree model requires $\Omega(\sqrt{n})$ queries, as shown by Goldreich and Ron [GR97]; therefore, for constant ε, a QCMAP with proof complexity $\tilde{O}(n^{1/4})$ is enough to overcome the testing lower bound, while the MAP of [GR18] requires a proof of length $\tilde{O}(\sqrt{n})$. Of course, a fairer comparison would be to *quantum* testers, for which $\tilde{O}(n^{1/3})$ queries suffice [Amb07] but no nontrivial lower bound is known. Thus, a QCMAP with proof length $\tilde{O}(\sqrt{n})$ can outperform the best known quantum tester.

6 Complexity separations

We now shift gears and begin to chart the landscape of complexity classes to which quantum proofs of proximity belong. In Section 6.1, we provide definitions that will be necessary in the remainder of the section, mainly pertaining to coding theory.

Our main goal is to prove Theorem 5, namely, that QMAPs can exploit quantum resources and the availability of a proof to gain expressivity that neither can provide separately. This theorem follows from the *incomparabilty* between the classes $\mathcal{MAP}$ and $\mathcal{QPT}$: in Section 6.2, we exhibit a property Π_B that is easy to test classically with a short proof, but requires many queries (without a proof) even for a quantum tester (Theorem 6.3); moreover, in Section 6.3, we show the existence of a property Π_F that is easily testable quantumly but difficult to test classically, even with the aid of a proof (Theorem 6.4).

The aforementioned results immediately imply the existence of a property, namely $\Pi_B \times \Pi_F$, which does not admit efficient MAPs nor quantum testers, requiring large proof or query complexity, whereas a QMAP with logarithmic proof and query complexities does exist (indeed, one with a *classical* proof).

Theorem 6.1 (Theorem 5, restated). *There exists a property $\Pi \subseteq \{0,1\}^n$ such that, for any small enough constant $\varepsilon > 0$,*

$$\Pi \in \mathcal{QCMAP}(\varepsilon, \log n, O(1))$$

and

$$\Pi \notin \mathcal{QPT}(\varepsilon, o(n^{0.49})) \cup \mathcal{MAP}(\varepsilon, p, q)$$

when $p \cdot q = o(n^{1/4})$.

6.1 Preliminaries

We first define the necessary notions of local codes that will be used in this section.

Definition 4 (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 $\mathbb{P}[T^w = 1] \geq 1 - \sigma$.*
2. *For every w that is ε-far from C, we have $\mathbb{P}[T^w = 0] \geq 1 - \sigma$.*

Note that the algorithm T that an LTC admits is simply an ε-tester for the property of being a valid codeword of C.

Definition 5 (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*

$$\mathbb{P}[D^w(i) = x_i] \geq 1 - \sigma.$$

Since the best known constructions of LDCs have superpolynomial blocklength, we will make use of a relaxation of this type of code that allows for much more efficient constructions and suffices for our purposes.

Definition 6 (Relaxed LDCs). *A code $C\colon \{0,1\}^k \to \{0,1\}^n$ is a q-local relaxed LDC with success rate ρ and decoding radius $\delta \in (0, \delta_C/2)$ 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 $x \in \{0,1\}^k$,*
$$\mathbb{P}[D^w(i) = x_i] \geq 2/3.$$
2. *Relaxed Decoding: For any $i \in [k]$ and any $w \in \{0,1\}^n$ that is δ-close to a (unique) codeword $C(x)$,*
$$\mathbb{P}[D^w(i) \in \{x_i, \perp\}] \geq 2/3.$$
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 $i \in I_w$,*
$$\mathbb{P}[D^w(i) = x_i] \geq 2/3\ .$$

As shown by [BGH$^+$06, GGK15, CGS20, AS20], there exist linear codes of only slightly super-linear blocklength that are both locally testable *and* relaxed locally decodable:

Theorem 6.2. *For any constant $\gamma > 0$, there exist linear codes with blocklength $n = k^{1+\gamma}$ that are locally testable and relaxed locally decodable with $O(1)$ queries, with respect to proximity parameter and decoding radius $\varepsilon, \delta = \Omega(1)$.*

Moreover, the tester and local decoder for these codes are one-sided (i.e., always accept when given a valid codeword as input), and the blocklength cannot be improved to linear [GL20, DGL21].

Communication complexity. In the model of quantum communcation complexity, two parties with unbounded computational power aim to compute a joint predicate by communicating the smallest number of qubits with each other. Alice knows $x \in \{0,1\}^k$, Bob knows $y \in \{0,1\}^k$ and both hold a function $f : \{0,1\}^{2k} \to \{0,1\}$, and, by communicating qubits with each other, they must compute $f(x,y)$ with bounded probability of error. The *communication complexity* of f is the worst-case number qubits that need to be communicated in order to compute $f(x,y)$ over all x,y, minimised over all communication protocols.

We will make use of the well-known communication complexity problem of *disjointness.*

Definition 7. *Let $x, y \in \{0,1\}^k$ and $S, T \subseteq [k]$ be sets whose indicator vectors are x and y, respectively; i.e., $S = \{i \in [k] : x_i = 1\}$ and $T = \{i \in [k] : y_i = 1\}$. Then $\mathsf{DISJ}_k(x,y) = 1$ if and only if S and T are disjoint, that is,*

$$\mathsf{DISJ}_k(x,y) = \begin{cases} 1 & \text{if } S \cap T = \varnothing \\ 0 & \text{otherwise} \end{cases} = \begin{cases} 1 & \text{if } x_i = 0 \text{ or } y_i = 0 \text{ for all } i \in [k], \\ 0 & \text{otherwise.} \end{cases}$$

This problem is known to be hard for quantum communication protocols, requiring $\Omega(n)$ qubits of communication, as shown in [Raz03].

6.2 MAPs versus quantum testers

We now set out to prove Theorem 6.3, which shows a property Π_B that is efficiently testable with a short classical proof (Lemma 3) but for which a quantum tester must make a large number of queries (Lemma 4).

The property in question is defined as follows. First, consider a linear code $C : \mathbb{F}^k \to \mathbb{F}^n$, where $\mathbb{F}$ is an extension field of $\mathbb{F}_2$ of degree $O(1)$ and $n = k^{1.001}$ that is both *locally testable* and *relaxed locally decodable* with $O(1)$ queries, proximity parameter $\varepsilon = \Omega(1)$ and decoding radius $\delta = \Omega(1)$ (recall that Theorem 6.2 shows that codes with these parameters exist).[14]

The property Π_B comprises the encoding of *non-Boolean* messages, that is:

$$\Pi_B = \left\{ C(z) : z \in \mathbb{F}^k \setminus \{0,1\}^k \right\} .$$

We first show that the property Π_B is efficiently testable via a MAP protocol with a short proof.

Lemma 3. *For any constant $\varepsilon \in (0, \delta]$, $\Pi_B \in \mathcal{MAP}(\varepsilon, \log n, O(1))$.*

[14]We note that while we define LTCs and RLDCs with respect to *binary* alphabets, they can be constructed over larger fields; see, e.g., [GGK19].

Proof. The verifier will follow the following strategy to test Π_B: first, test whether the input is close to the code C (which can be accomplished with $O(1)$ queries due to the local testability of C). If the tester rejects, then not only is the input far from Π_B, but from all of C, in which case it rejects.

Except with small probability, if the tester accepts the input is close to C, so we may locally decode any coordinate of the message (also with $O(1)$ queries); using the proof string to determine this location, the verifier then checks if the symbol at that coordinate is boolean-valued.

This strategy is laid out in Algorithm 1.

Algorithm 1: MAP verifier for Π_B

```
Input: explicit access to a proximity parameter $\varepsilon > 0$ and a proof string $\pi \in \{0,1\}^{\log n}$, as
       well as oracle access to $x \in \mathbb{F}^n$.
1 Test if the input $w$ is a valid codeword with proximity parameter $\varepsilon$. Reject if the test
   rejects.
2 Interpret the proof as an index $i \in [k]$, locally decode $z_i$ and accept if $z_i \in \mathbb{F} \setminus \{0,1\}$.
   Otherwise, reject.
```

Note that the proof complexity is $\log n$ by definition, and, since both local testing and local decoding have query complexity $O(1)$, the verifier makes $O(1)$ queries in total (note that querying an element of $\mathbb{F}$ requires $O(1)$ *bit* queries, so the complexities of the tester and decoder are still constant in terms of bit queries). Moreover, if $w \in \Pi_B$, then $w = C(z)$ for some $z \in \mathbb{F}^k \setminus \{0,1\}^k$, and local testing succeeds with probability 1; and when the prover specifies a coordinate i such that $z_i \notin \{0,1\}$, local decoding also succeeds with probability 1, so completeness follows.

Now, if w is ε-far from Π_B, then either (1) the input w is ε-far from *any* codeword of the code C; or (2) w is ε-close to some $C(z)$ such that $z \in \{0,1\}^k$.

In the first case, local testing (and thus the verifier) will reject with probability 2/3. In the second case, the testing step may not trigger a rejection, but the local decoder then outputs, regardless of the proof $i \in [k]$, either $\perp$ or $z_i \in \{0,1\}$ with probability 2/3, any of which cause the verifier to reject. □

The next lemma shows that, unlike MAPs, quantum testers cannot test Π_B efficiently.

Lemma 4. *Any quantum tester for the property Π_B with constant proximity parameter $\varepsilon \in [0, \delta)$ must have query complexity $\Omega(n^{0.49})$.*

Proof. Recall that in the disjointness problem, Alice is given as input $x \in \{0,1\}^k$, Bob is given $y \in \{0,1\}^k$ and they must compute $\mathsf{DISJ}_k(x,y)$; and that Π_B is the encoding of non-Boolean strings of length $k = n^{\frac{1}{1.001}}$ by the code C.

To show that any quantum tester needs $\Omega(n^{0.49})$ queries to ε-test Π_B, we give a reduction showing that a tester with query complexity q can be used to compute DISJ_k by communicating $O(q \log n)$ qubits. Since the quantum communication complexity of DISJ_k is $\Omega(\sqrt{k}) = \Omega(n^{\frac{1}{2}\cdot\frac{1}{1.001}}) = \Omega(n^{0.499})$, the query complexity of the quantum tester follows.

First, Alice and Bob use C to encode x and y, respectively. Now Alice holds $C(x) \in \mathbb{F}^n$ and Bob holds $C(y) \in \mathbb{F}^n$. Note that, defining $z := x + y \in \mathbb{F}^k$, we have $\mathsf{DISJ}(x,y) = 0 \iff z \notin \{0,1\}^k \iff C(z) \in \Pi_B$. Now, Alice and Bob respectively set up the unitaries U_A and U_B shown below, where the first register holds $\log n$ qubits and the second holds $O(1)$ qubits (enough to specify a single element of $\mathbb{F}$).

$$\forall i \in [n], \alpha \in \mathbb{F}, \quad \begin{aligned} U_A \left|i\right\rangle \left|\alpha\right\rangle &= \left|i\right\rangle \left|\alpha + C(x)_i\right\rangle \\ U_B \left|i\right\rangle \left|\alpha\right\rangle &= \left|i\right\rangle \left|\alpha + C(y)_i\right\rangle \ . \end{aligned}$$

Note that, since $\mathbb{F}_2 \subset \mathbb{F}$, the sums are linear operations over $\mathbb{F}_2$; that is, if $C(x)_i = c_1 c_2 \dots c_\ell$ and $\alpha = \alpha_1 \alpha_2 \dots \alpha_\ell$ as bit strings, then $\left|\alpha + C(x)_i\right\rangle = \left|c_1 \oplus \alpha_1\right\rangle \left|c_2 \oplus \alpha_2\right\rangle \cdots \left|c_\ell \oplus \alpha_\ell\right\rangle$ (note that this equality is the only place where we use the fact that $\mathbb{F}$ is an extension of $\mathbb{F}_2$). Alice simulates the quantum tester and only communicates with Bob in order to make a query to the oracle; if the tester accepts, Alice outputs 0, and she outputs 1 otherwise.

More precisely, whenever the tester calls the oracle U, which acts as $U \left|i\right\rangle \left|\alpha\right\rangle = \left|i\right\rangle \left|\alpha + C(z)_i\right\rangle$ on a $(\log n + O(1))$-bit quantum state ρ, Alice first applies U_A to ρ, then sends all qubits to Bob; Bob then applies U_B on the qubits it receives and returns them to Alice. This communicates a total of $O(\log n)$ qubits and implements the same transformation as querying U, since $U_B \cdot U_A \left|i\right\rangle \left|j\right\rangle = \left|i\right\rangle \left|j + C(x)_i + C(y)_i\right\rangle = \left|i\right\rangle \left|j + C(x+y)_i\right\rangle$ by the linearity of the code C and the fact that $z = x+y$.

Each query made by the tester entails $O(\log n)$ qubits of communication, so that after q queries, Alice and Bob exchange $O(q \cdot \log n)$ qubits in total. The tester accepts with probability at least $2/3$ when $C(z) \in \Pi_B \iff \mathsf{DISJ}(x,y) = 0$, in which case Alice outputs 0. If $\mathsf{DISJ}(x,y) = 1$, we have that $C(z)$ is δ-far from Π_B (since the relative distance of C is δ). Since $\varepsilon \leq \delta$, the ε-tester rejects with probability $2/3$ and Alice outputs 1 in this case.

Thus, Alice is able to compute DISJ_k with $O(q \cdot \log n)$ qubits of communication. Since the quantum communication complexity of DISJ_k is $\Omega(\sqrt{k})$ [Raz03], we conclude that the tester must make $\Omega(\sqrt{k}/\log n) = \Omega(n^{0.499}/\log n) = \Omega(n^{0.49})$ queries. □

Remark 4. Although we reduce DISJ to testing *non*-Booleanity, a symmetric argument shows the same lower bound for the (arguably more natural) property of *Booleanity* $\left\{C(z) : z \in \{0,1\}^k\right\}$. Often, one key step in PCP constructions is to check that an encoding corresponds to a logical assignment, i.e., that it is the encoding of a Boolean message. Therefore, bounds on Booleanity may have consequences for PCPs.

We conclude this section with the separation immediately implied by Lemma 3 and Lemma 4.

Theorem 6.3. *For every constant $\varepsilon \in (0, \delta]$, the property Π_B belongs to $\mathcal{MAP}(\varepsilon, \log n, O(1))$ but does not belong to $\mathcal{QPT}(\varepsilon, o(n^{0.49}))$. Therefore,*

$$\mathcal{MAP}(\varepsilon, \log n, O(1)) \not\subseteq \mathcal{QPT}(\varepsilon, o(n^{0.49})) \ .$$

6.3 Quantum testers versus MAPs

In this section, we will show the existence of a property that are easily testable with a quantum tester, but for which a classical tester – even with additional access to a proof – must make a number of queries that depends strongly on the length of the input. More formally, we will show in Theorem 6.4 the existence of a property of n-bit strings in $\mathcal{QPT}(\varepsilon, O(1/\varepsilon))$ that is not in $\mathcal{MAP}(\varepsilon, p, q)$ when $p \cdot q = o(n^{1/4})$ and ε is a small enough constant.

The property in question is derived from forrelation, a problem that strongly separates classical and quantum algorithms in the query model; in fact, the work that proved such a separation already shows that it carries over to the property testing setting [AA18], which we will extend to the setting of MAPs. Formally, we have

Lemma 5 ([AA18]). *Define the property* Π_F *as*

$$\Pi_F = \{(f,g) : \Phi_{f,g} \leq 1/100\}\,,$$

where (f,g) *are* $n/2$*-bit strings corresponding to pairs of* $\log(n/2)$*-bit Boolean functions and* $\Phi_{f,g} = (n/2)^{-3/2} \sum_{x,y\in\{0,1\}^{\log(n/2)}} f(x)(-1)^{x\cdot y} g(y)$.

Then, for any $\varepsilon > 0$ *sufficiently small,* $\Pi_F \in \mathcal{QPT}(\varepsilon, O(1/\varepsilon))$ *and* $\Pi_F \notin \mathcal{PT}(\varepsilon, o(\sqrt{n}/\log n))$.

Therefore, the property Π_F is easy for quantum testers and hard for their classical counterparts. This section is thus devoted to showing that testing Π_F is hard not only for property testers, but for MAPs as well: we will prove that a MAP for Π_F requires proof length p and query complexity q satisfying $pq = \Omega(n^{1/4})$. We first introduce relevant definitions and theorems, then describe the steps of the proof.

Recall that $\mathcal{MA}$ is the class of languages that are decidable *in polynomial time* with a (polynomial-size) proof string, the analogue of which is $\mathcal{MAP}$ in the property-testing setting. By [HHT93], $\mathcal{MA}$ is contained in the class $\mathcal{BPP}_{\text{path}}$ of languages decidable (with high probability) by a randomised Turing machine whose computational paths are all equally likely.[15] Query lower bounds for forrelation (i.e., deciding whether $|\Phi_{f,g}| \leq 1/100$ or $\Phi_{f,g} > 3/5$ for a pair (f,g) of Boolean functions) against the latter are known:

Proposition 3 ([Aar10, Che16]). *Any* BPP_{path} *algorithm for forrelation must make* $\Omega(n^{1/4})$ *queries to its input.*

We are now ready to describe the three steps taken in proving hardness of Π_F for $\mathcal{MAP}$: we (1) show that transforming an MA algorithm with proof complexity p and query complexity q into a BPP_{path} one [HHT93] yields an algorithm with query complexity $O(pq)$; (2) show how a MAP for Π_F implies an $\mathcal{MA}$ algorithm with the same parameters *for forrelation*; and (3) conclude that $pq = o(n^{1/4})$ implies a $\mathcal{BPP}_{\text{path}}$ upper bound of $O(pq) = o(n^{1/4})$ for the query complexity of forrelation, which contradicts Proposition 3.

The original proof of $\mathcal{MA} \subseteq \mathcal{BPP}_{\text{path}}$ ([HHT93], Theorem 3.7) takes an MA algorithm with proof complexity p and constant success probability, repeats the execution $O(p)$ times, amplifying the success probability to $1 - O(2^{-p})$, and then defines a BPP_{path} machine as follows. The machine (non-deterministically) guesses a proof string and simulates the MA algorithm with it, spawning "dummy" execution paths if the MA algorithm accepts. Inspecting this transformation in the query model, we obtain a quadratic overhead: if the MA algorithm has query complexity q and proof complexity p, the BPP_{path} machine thus obtained has query complexity $O(pq)$ (the $O(p)$ repetitions of the MA algorithm increase its query complexity multiplicatively by this amount, while the dummy paths make no queries).

The second step is formalised by the following lemma.

Lemma 6. *A MAP protocol for* Π_F *with sufficiently small proximity parameter* $\varepsilon > 0$ *implies an* $\mathcal{MA}$ *algorithm for forrelation with the same query and proof complexities as the MAP protocol.*

Proof. We define an MA protocol *for forrelation* (as a gap problem) the natural way: the proofs and queries correspond to the proofs and queries of the MAP, and the MA verifier accepts if and only if the MAP rejects.

[15] In $\mathcal{BPP}$, the probability of following a computational path is a function of its length (which coincides with the number of random coins flipped by the algorithm). $\mathcal{BPP}_{\text{path}}$ differs from $\mathcal{BPP}$ by lifting this restriction.

To show correctness of this protocol, we follow the reduction of [AA18]. Specifically, [AA18, Lemma 40] shows that any (f', g') such that f' is ε-close to f and g' is ε-close to g satisfies $|\langle f', Hg'\rangle - \langle f, Hg\rangle| = O(\sqrt{\varepsilon}\log(1/\varepsilon))$. By choosing a suitably small ε, the right-hand side is at most (say) $1/100$. Thus any (f, g) such that $\langle f, Hg\rangle > 3/5$ is ε-far from Π, and it follows that the MAP protocol will accept (with high probability) pairs (f, g) such that $|\langle f, Hg\rangle| \leq 1/100$ and will reject if $\langle f, Hg\rangle > 3/5$. Therefore, the MA protocol is able to distinguish between the two cases. □

A simple argument now proves the separation.

Theorem 6.4. *Let $\varepsilon > 0$ be a small enough constant. There exists a property Π_F such that $\Pi_F \in \mathcal{QPT}(\varepsilon, O(1/\varepsilon))$ and $\Pi_F \notin \mathcal{MAP}(\varepsilon, p, q)$ for any p, q such that $p \cdot q = o(n^{1/4})$.*

Proof. Suppose, towards contradiction, that there existed a MAP for Π_F with any proximity parameter ε, as well as proof complexity p and query complexity q satisfying $p \cdot q = o(n^{1/4})$.

By Lemma 6, there exists an MA protocol for forrelation with the same query and proof complexities, which can then be transformed into a BPP_{path} algorithm with query complexity $O(p \cdot q) = o(n^{1/4})$ for the same problem. But this contradicts the $\Omega(n^{1/4})$ lower bound of Proposition 3. □

We are finally ready to prove the main separation, by exhibiting a property Π in $\mathcal{QMAP}$ which is in neither $\mathcal{MAP}$ nor $\mathcal{QPT}$.

Proof of Theorem 6.1. Recall that our goal is to show that the property $\Pi = \Pi_B \times \Pi_F$ that is efficiently ε-testable by a QMAP, but not by a quantum tester (without a proof) nor classicaly with a proof, for some small enough $\varepsilon = \Omega(1)$. To this end, we invoke Theorem 6.3 and Theorem 6.4 and give the following verifier strategy explicitly. Note that, while the strings in Π_B are over an alphabet $\mathbb{F}$ larger than $\{0, 1\}$, since $|\mathbb{F}| = O(1)$ each symbol can be represented by $O(1)$ bits (and the proof indicates the first bit in such a block).

```
Algorithm 2: QCMAP verifier for $\Pi_B \times \Pi_F$
  Input: explicit access to a proximity parameter $\varepsilon' = 2\varepsilon > 0$ and a proof $i \in [n/2]$, as well
         as oracle access to a string $z \in \{0,1\}^n$
1 Interpret the input as a concatenation of $n/2$-bit strings $x$ and $y$. Use the algorithm of
   Lemma 3 to verify, with $O(1)$ queries and the proof $i$, whether $x \in \Pi_B$ with proximity
   $\varepsilon := \varepsilon'/2$.
2 Use the quantum tester of Lemma 5 to test, with $O(1)$ queries, if $y \in \Pi_F$ with proximity $\varepsilon$.
3 If both of the previous tests accepted, then accept; otherwise, reject.
```

Completeness follows immediately from Lemma 3 and Lemma 5, since the verifier for Π_B accepts with certainty and the and tester for Π_F accepts with probability $2/3$ when $x \in \Pi_B$ and $y \in \Pi_F$. If, on the other hand, (x, y) is ε-far from $\Pi_B \times \Pi_F$, then either x is ε-far from Π_B or y is ε-far from Π_F, and either the verifier for Π_B or the tester for Π_F will reject (with probability $2/3$). Thus, the QCMAP verifier for Π (executed with respect to proximity parameter $\varepsilon' = 2\varepsilon$) implies $\Pi \in \mathcal{QCMAP}(\varepsilon, \log n, O(1))$.

All that remains is to show $\Pi = \Pi_B \times \Pi_F$ does not admit an efficient quantum tester nor a MAP. Assume, towards contradiction, that either $\Pi \in \mathcal{QPT}(\varepsilon, o(n^{0.49}))$ or $\Pi \in \mathcal{MAP}(\varepsilon, p, q)$ when

$p \cdot q = o(n^{1/4})$. In the first case, applying the tester for Π to $\Pi_B \times \{y\}$ for some fixed $y \in \Pi_F$ shows that $\Pi_B \in \mathcal{QPT}(\varepsilon, o(n^{0.49}))$, a contradiction with Lemma 4. In the second case, applying the MAP protocol for Π to $\{x\} \times \Pi_F$, for some fixed $x \in \Pi_B$, shows that $\Pi_F \in \mathcal{MAP}(\varepsilon, p, q)$, a contradiction with Theorem 6.4. □

7 A hard class of problems for QMAPs

When introducing a new complexity class in the landscape of known classes, it is important not only to exhibit problems it can solve, but also problems it cannot. We set out to show a natural limitation on QMAPs in this section, by answering (negatively) the following question: if a property "looks random" on any subset of q coordinates, can a quantum proof be of any help to a verifier with query complexity q? Intuitively, the answer should be no: if querying q coordinates provides no information as to whether or not an input satisfies a property, then any proof (quantum or otherwise) should not be able to offer more information in conjunction with the queries than it does on its own.

We formalise this intuition in Theorem 7.2, which states the following: if a property $\Pi \subset \{0,1\}^n$ is k-wise independent and sparse (i.e., its size $|\Pi|$ is sufficiently small compared to the set of all 2^n bit strings), then k is a lower bound on the number of queries made by randomized query algorithm that accepts all inputs in Π with probability strictly greater than 1/2, and rejects with probability strictly greater than 1/2 when run on any input that is far from Π. In other words, the $\mathcal{UPP}$ query complexity of testing Π is at least k (recall that $\mathcal{UPP}$ is the query model version of $\mathcal{PP}$, which captures randomized computation with small bias). Note that *some* assumption on the sparsity of Π is necessary for any non-trivial lower bound to hold, if only to rule out, e.g., the trivially testable property $\Pi = \{0,1\}^n$.

Combining Theorem 7.2 with the well-known inclusion $\mathcal{QMA} \subseteq \mathcal{PP}$ [MW05] allows us to conclude the following: for any k-wise independent and sufficiently sparse property Π, the product of proof and query complexities of a QMAP for verifying membership in Π with constant proximity parameter ε is $\Omega(k)$ (see Corollary 9).

The proof of Theorem 7.2 works as follows. Our analysis shows that sparsity of Π ensures there exists a subset $\Pi' \subset \{0,1\}^n$ that is far from Π such that Π' is *also* k-wise independent (see Lemma 7). This means that any query algorithm making fewer than k queries cannot distinguish a random input in Π from a random input in Π', as both sets "look random" when inspecting only k bits of a randomly chosen input from the set. Yet since Π' is far from Π, any testing procedure for Π must distinguish Π from Π'. Hence, any tester for Π must make k queries (even if it only outputs the correct answer on inputs in Π and Π' with probability strictly greater than 1/2).

Technical Details. We begin recalling the definition of k-wise independence.

Definition 8. *A set of strings $S \subseteq \{0,1\}^n$ is called k-wise independent if, for any fixed set of indices $I \subset [n]$ of size k, the string $x_{|I}$ is uniformly random when x is sampled uniformly from S. Equivalently, for every $y \in \{0,1\}^k$,*

$$\left|\{x \in S : x_{|I} = y\}\right| = \frac{|S|}{2^k}.$$

We next show that, given any small enough set S of strings, there exists a $\Omega(n)$-wise independent set that is ε-far from S. In the following lemma, H denotes the binary entropy function $H(\alpha) = -\alpha \log \alpha - (1-\alpha)\log(1-\alpha)$ (whose restriction to $[0,1/2]$ is bijective).

Lemma 7. *Let $\varepsilon \in (0, H^{-1}(1/4))$ and $S \subseteq \{0,1\}^n$ be such that $|S| < 2^{(1/4-H(\varepsilon))n}$. Then there exists a linear code C that is ε-far from S with dual distance $\Omega(n)$; equivalently, C is $\Omega(n)$-wise independent.*

Proof. Let $C\colon \{0,1\}^{3n/4} \to \{0,1\}^n$ be a random linear code (where each entry of its generator matrix is a Bernoulli(1/2) random variable). Then, for every $x \in \{0,1\}^{3n/4}$, the codeword $C(x)$ is uniformly random in $\{0,1\}^n$ (but *not* independent of other codewords). Denoting by $N_\varepsilon(S)$ the ε-neighbourhood of S (i.e., the set of bit strings at distance at most ε from S), we have:

$$\begin{aligned}\mathbb{P}[C \cap N_\varepsilon(S) \neq \varnothing] &\leq \sum_{x \in \{0,1\}^{3n/4}} \mathbb{P}[C(x) \in N_\varepsilon(S)] = 2^{3n/4} \cdot \frac{|N_\varepsilon(S)|}{2^n} \\ &\leq \frac{|S| \cdot 2^{H(\varepsilon)n}}{2^{n/4}} < 1,\end{aligned}$$

and, by the probabilistic method, there exists a code $C \subset \{0,1\}^n$ of size $2^{3n/4}$ that is ε-far from S. Moreover, the dual code $C^\perp\colon \{0,1\}^{n/4} \to \{0,1\}^n$ is a linear code whose distance meets the Gilbert-Varshamov bound with high probability; that is, the distance of this dual code is $\Omega(n)$ with probability $1 - o(1)$, proving the claim. □

The previous lemma, when applied to a "random-looking" set S (i.e., a k-wise independent S, for $k = o(n)$), will ensure that S and the code C are hard to distinguish. To make this precise, we first recall the definition of the *threshold degree* of a (partial) function.

Definition 9. *Let $\mathcal{X} \subseteq \{1,-1\}^n$ and let $f : \mathcal{X} \to \{1,-1\}$ be any function defined on domain $\mathcal{X} \subseteq \{1,-1\}^n$.*[16] *The threshold degree of f, denoted $\mathsf{thrdeg}(f)$, is the minimal degree of an n-variate polynomial p that sign-represents f, i.e., such that $f(x) = \mathsf{sgn}(p(x))$ for all $x \in \mathcal{X}$.*[17] *Note that no constraints are placed on the behaviour of $p(x)$ at inputs in $\{1,-1\}^n \setminus \mathcal{X}$.*

The threshold degree is a measure of complexity of Boolean functions (in particular), so that we expect functions with high threshold degree to also have high query complexity. This intuition is validated by the following folklore result: the minimal query complexity of a UPP algorithm that computes f is exactly equal to its threshold degree. We provide a proof of this fact for completeness, as, to the best of our knowledge, it is not explicitly proven in the literature.

We write $f \in \mathcal{UPP}(q)$ when there exists a UPP algorithm with query complexity q that computes f, and denote by $q(f)$ the integer such that $f \in \mathcal{UPP}(q(f))$ but $f \notin \mathcal{UPP}(q(f)-1)$.

Lemma 8. *For any $\mathcal{X} \subseteq \{1,-1\}^n$ and $f\colon \mathcal{X} \to \{1,-1\}$, it holds that $\mathsf{thrdeg}(f) = q(f)$.*

Proof. We prove both inequalities, starting with $q(f) \leq \mathsf{thrdeg}(f) \coloneqq d$.

Let $P(X_1,\ldots,X_n) = \sum_{S \subset [n], |S| \leq d} \alpha_S \prod_{i \in S} X_i$ be a polynomial of degree d that sign-represents f, i.e., such that $f(x) = \mathsf{sgn}(P(x))$ for all $x \in \mathcal{X}$. Consider the algorithm A (with query complexity

[16] For notational convenience, we consider Boolean functions with codomain $\{1,-1\}$, noting that this is equivalent to the usual codomain $\{0,1\}$ by mapping $0 \to 1$, $1 \to -1$, and $\oplus$ to multiplication.

[17] Here, $\mathsf{sgn}(t)$ is defined to equal 1 if $t > 0$, -1 if $t < 0$, and 0 if $t = 0$.

d) that queries the set of coordinates S with probability $|\alpha_S|/\sum_{|T|\le d}|\alpha_T|$ and outputs $\mathsf{sgn}(\alpha_S)\cdot\prod_{i\in S}x_i$. Fix $x\in\mathcal{X}$ and suppose, without loss of generality, that $f(x)=1$. We thus have

$$\mathbb{E}[A^x]=\frac{1}{\sum_{|T|\le d}|\alpha_T|}\sum_{|S|\le d}|\alpha_S|\cdot\mathsf{sgn}(\alpha_S)\cdot\prod_{i\in S}x_i=\frac{P(x)}{\sum_{|T|\le d}|\alpha_T|}>0,$$

and, since A^x only outputs 1 or -1, we have $\mathbb{P}[A^x=-1]+\mathbb{P}[A^x=1]=1$ and thus $\mathbb{P}[A^x=1]>1/2$. It follows that A is a UPP algorithm for f with query complexity d and thus $q(f)\le\mathsf{thrdeg}(f)$.

To prove the reverse inequality, consider a UPP algorithm A that computes f with query complexity $q\coloneqq q(f)$, given by a distribution over decision trees of depth at most q. To see that the function computed by each decision tree T can be sign-represented by a polynomial of degree at most q (which is a standard fact), we follow the exposition on *leaf indicators* in [GM21]. Denote by L the set of leaves of T, and identify each $\ell\in L$ with its indicator function $\ell\colon\{1,-1\}^n\to\{0,1\}$ such that $\ell(x)=1$ if and only if ℓ is the unique leaf reached on input x in T.

Then, if $c_\ell\in\{1,-1\}$ is the output of the decision tree when an execution ends at the leaf ℓ, the output of T on input x is $\sum_{\ell\in L}c_\ell\cdot\ell(x)$. Thus, showing $\ell(\cdot)$ can be represented by a polynomial of degree at most q implies the same degree bound for the computation of T. Fix $\ell\in L$, let $(i_1,\ldots,i_d)\in[n]^q$ be the coordinates queried by the root-to-leaf path that ends at ℓ, and let the sequence of bits $(b_1^\ell,\ldots,b_d^\ell)\in\{1,-1\}^q$ correspond to the queried values that cause this path to be followed. Then,

$$\ell(x_1,\ldots,x_n)=\prod_{j=1}^{q}\frac{x_{i_j}+b_j}{2b_j},$$

so $\ell(\cdot)$ can be represented by the degree-q polynomial $P(X_1,\ldots,X_n)=2^{-q}\prod_{j=1}^{q}(X_{i_j}+b_j)/b_j$. Thus, the output of A^x *when it selects this tree* is the degree-q polynomial $\sum_{\ell\in L}c_\ell\cdot\ell(x)$, and $\mathbb{E}[A^x]$ is a convex combination of such sums (which also has degree q). Since $f(x)=\mathsf{sgn}(\mathbb{E}[A^x])$ for all $x\in\mathcal{X}$, we conclude that $\mathsf{thrdeg}(f)\le q(f)$ and the claim follows. $\square$

The final ingredient to show the lower bound is the next theorem, a special case of the "Theorem of the Alternative" [OS03, ABFR91].

Theorem 7.1. *Let $\mathcal{X}\subseteq\{1,-1\}^n$ and let $f\colon X\to\{1,-1\}$ be any partial Boolean function defined over domain $\mathcal{X}$. If there exists a distribution $\mathcal{D}$ on $\mathcal{X}$ such that $\mathbb{E}_{x\leftarrow\mathcal{D}}[f(x)\cdot m(x)]=0$ for every monomial m of degree less than k, then the threshold degree of f is at least k.*

We are now ready to prove the main result of this section.

Theorem 7.2. *Let $\Pi\subseteq\{1,-1\}^n$ be a k-wise independent property such that $|\Pi|<2^{(1/4-H(\varepsilon))n}$ with $k=o(n)$. Then $f\notin\mathcal{UPP}(k-1)$, where f is the partial function such that $f(x)=-1$ when $x\in\Pi$, $f(x)=1$ when x is ε-far from Π, and f is undefined otherwise.*

Proof. First, apply Lemma 7 to obtain a k-wise independent code $C\subseteq\{0,1\}^n$ that is ε-far from Π. Let $\mathcal{D}$ be the distribution obtained by drawing a uniform random element of Π with probability $1/2$ and drawing a uniform random element of C with probability $1/2$. Then for every monomial m of degree less than k,

$$\mathbb{E}_{x\leftarrow\mathcal{D}}[f(x)m(x)]=\frac{\mathbb{E}_{x\leftarrow\Pi}[f(x)m(x)]+\mathbb{E}_{x\leftarrow C}[f(x)m(x)]}{2}=\frac{\mathbb{E}_{x\leftarrow\Pi}[m(x)]-\mathbb{E}_{x\leftarrow C}[m(x)]}{2}=0$$

The final equality above holds by virtue of the k-wise independence of both Π and C. Let $\mathcal{X}$ be the union of inputs in Π and inputs that are ε-far from Π. Define the partial function f over domain $\mathcal{X}$ via:

$$f(x) = \begin{cases} -1 & \text{, if } x \in \Pi \\ 1 & \text{, if } x \in \mathcal{X} \setminus \Pi. \end{cases}$$

By Theorem 7.1, the distribution $\mathcal{D}$ constructed above witnesses the fact that $\mathsf{thrdeg}(f) \geq k$. Since the $\mathcal{UPP}$ query complexity of f is $\mathsf{thrdeg}(f)$ by Lemma 8, the claim follows. $\square$

We conclude the section with a corollary that follows from the inclusion $\mathcal{QMA} \subseteq \mathcal{PP}$. The proof of this inclusion (in the polynomial-time setting) proceeds in two steps: (1) reducing the error rate of a QMA algorithm to roughly 2^{-p}, where p is the length of the proof given to the verifier, by repeating the algorithm $O(p)$ times; and (2) running the verifier with the proof fixed to be the maximally mixed state. This exhibits a gap of roughly 2^{-p} between the acceptance probabilities of yes- and no-inputs, which suffices to place the problem in $\mathcal{PP}$ [MW05, Wat09]. The same transformation, carried out *in the query model*, implies that any sufficiently small $\Pi \in \mathcal{QMAP}(\varepsilon, p, q)$ can be "ε-tested" by a UPP algorithm with query complexity $O(pq)$; that is, any function f as in the statement of Theorem 7.2 is such that $f \in \mathcal{UPP}(O(pq))$. Therefore,

Corollary 9. *For any sufficiently constant small $\varepsilon > 0$ and k-wise independent property $\Pi \subseteq \{0,1\}^n$ such that $|\Pi| < 2^{n/5}$, we have $\Pi \notin \mathcal{QMAP}(\varepsilon, p, q)$ unless $pq = \Omega(k)$.*

References

[AA18] Scott Aaronson and Andris Ambainis. Forrelation: A problem that optimally separates quantum from classical computing. *SIAM Journal on Computing*, 47(3):982–1038, 2018.

[Aar10] Scott Aaronson. BQP and the polynomial hierarchy. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 141–150, 2010.

[Aar12] Scott Aaronson. Impossibility of succinct quantum proofs for collision-freeness. *Quantum Info. Comput.*, 12(1–2):21–28, January 2012.

[ABFR91] James Aspnes, Richard Beigel, Merrick L. Furst, and Steven Rudich. The expressive power of voting polynomials. In Cris Koutsougeras and Jeffrey Scott Vitter, editors, *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 402–409. ACM, 1991.

[ABRW16] Andris Ambainis, Aleksandrs Belovs, Oded Regev, and Ronald de Wolf. Efficient quantum algorithms for (gapped) group testing and junta testing. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 903–922. SIAM, 2016.

[ACL11] Andris Ambainis, Andrew M. Childs, and Yi-Kai Liu. Quantum property testing for bounded-degree graphs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 365–376, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[AK07] Scott Aaronson and Greg Kuperberg. Quantum versus classical proofs and advice. In *22nd Annual IEEE Conference on Computational Complexity (CCC 2007), 13-16 June 2007, San Diego, California, USA*, pages 115–128. IEEE Computer Society, 2007.

[Amb07] Andris Ambainis. Quantum walk algorithm for element distinctness. *SIAM Journal on Computing*, 37(1):210–239, 2007.

[AS20] Vahid R Asadi and Igor Shinkar. Relaxed locally correctable codes with improved parameters. *arXiv preprint arXiv:2009.07311*, 2020.

[BBM12] Eric Blais, Joshua Brody, and Kevin Matulef. Property testing lower bounds via communication complexity. *computational complexity*, 21(2):311–358, 2012.

[BCL20] Sebastien Bubeck, Sitan Chen, and Jerry Li. Entanglement is necessary for optimal quantum property testing. *arXiv preprint arXiv:2004.07869*, 2020.

[BFNR08] Harry Buhrman, Lance Fortnow, Ilan Newman, and Hein Röhrig. Quantum property testing. *SIAM Journal on Computing*, 37(5):1387–1400, 2008.

[BGH$^+$06] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. Robust pcps of proximity, shorter pcps, and applications to coding. *SIAM Journal on Computing*, 36(4):889–974, 2006.

[BHMT02] Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74, 2002.

[BOW19] Costin Bădescu, Ryan O'Donnell, and John Wright. Quantum state certification. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 503–514, 2019.

[BRV18] Itay Berman, Ron D Rothblum, and Vinod Vaikuntanathan. Zero-knowledge proofs of proximity. In *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[BSCG$^+$17] Eli Ben-Sasson, Alessandro Chiesa, Ariel Gabizon, Michael Riabzev, and Nicholas Spooner. Interactive oracle proofs with constant rate and query complexity. In *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[CFMDW10] Sourav Chakraborty, Eldar Fischer, Arie Matsliah, and Ronald De Wolf. New results on quantum property testing. *arXiv preprint arXiv:1005.0523*, 2010.

[CGG$^+$19] Clément L Canonne, Elena Grigorescu, Siyao Guo, Akash Kumar, and Karl Wimmer. Testing k-monotonicity: The rise and fall of boolean functions. *Theory of Computing*, 15(1):1–55, 2019.

[CGS20] Alessandro Chiesa, Tom Gur, and Igor Shinkar. Relaxed locally correctable codes with nearly-linear block length and constant query complexity. *31st ACM-SIAM Symposium on Discrete Algorithms*, 2020.

[Che00] Anthony Chefles. Quantum state discrimination. *Contemporary Physics*, 41(6):401–424, 2000.

[Che16] Lijie Chen. A note on oracle separations for BQP, 2016.

[CM13] Kaushik Chakraborty and Subhamoy Maitra. Improved quantum test for linearity of a boolean function. *arXiv preprint arXiv:1306.6195*, 2013.

[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.

[DLW$^+$21] Severin Daiss, Stefan Langenfeld, Stephan Welte, Emanuele Distante, Philip Thomas, Lukas Hartung, Olivier Morin, and Gerhard Rempe. A quantum-logic gate between distant quantum-network modules. *Science*, 371(6529):614–617, 2021.

[EHW$^+$20] Jens Eisert, Dominik Hangleiter, Nathan Walk, Ingo Roth, Damian Markham, Rhea Parekh, Ulysse Chabaud, and Elham Kashefi. Quantum certification and benchmarking. *Nature Reviews Physics*, pages 1–9, 2020.

[EKR04] Funda Ergün, Ravi Kumar, and Ronitt Rubinfeld. Fast approximate probabilistically checkable proofs. *Information and Computation*, 189(2):135–159, 2004.

[FGL14] Eldar Fischer, Yonatan Goldhirsh, and Oded Lachish. Partial tests, universal tests and decomposability. In *Proceedings of the 5th conference on Innovations in theoretical computer science*, pages 483–500, 2014.

[GG16] Oded Goldreich and Tom Gur. Universal locally verifiable codes and 3-round interactive proofs of proximity for CSP. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 23, page 192, 2016.

[GGK15] Oded Goldreich, Tom Gur, and Ilan Komargodski. Strong locally testable codes with relaxed local decoders. In *30th Conference on Computational Complexity, CCC 2015, June 17-19, 2015, Portland, Oregon, USA*, 2015.

[GGK19] Oded Goldreich, Tom Gur, and Ilan Komargodski. Strong locally testable codes with relaxed local decoders. *ACM Transactions on Computation Theory (TOCT)*, 11(3):1–38, 2019.

[GGL$^+$00] Oded Goldreich, Shafi Goldwasser, Eric Lehman, Dana Ron, and Alex Samorodnitsky. Testing monotonicity. *Combinatorica*, 20(3):301–337, 2000.

[GGR18] Oded Goldreich, Tom Gur, and Ron D Rothblum. Proofs of proximity for context-free languages and read-once branching programs. *Information and Computation*, 261:175–201, 2018.

[GL19] András Gilyén and Tongyang Li. Distributional property testing in a quantum world. *arXiv:1902.00814*, 2019.

[GL20] Tom Gur and Oded Lachish. On the power of relaxed local decoding algorithms. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1377–1394. SIAM, 2020.

[GLM08] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum random access memory. *Physical review letters*, 100(16):160501, 2008.

[GLR18] Tom Gur, Yang P Liu, and Ron D Rothblum. An exponential separation between MA and AM proofs of proximity. In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[GM21] Mika Göös and Gilbert Maystre. A majority lemma for randomised query complexity. *Electron. Colloquium Comput. Complex.*, 28:24, 2021.

[Gol14] Oded Goldreich. On multiple input problems in property testing. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2014)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2014.

[Gol17] Oded Goldreich. *Introduction to property testing.* Cambridge University Press, 2017.

[Gol20] Oded Goldreich. On the communication complexity methodology for proving lower bounds on the query complexity of property testing. In *Computational Complexity and Property Testing - On the Interplay Between Randomness and Computation*, volume 12050 of *Lecture Notes in Computer Science*, pages 87–118. Springer, 2020.

[GR97] Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 406–415, 1997.

[GR99] Oded Goldreich and Dana Ron. A sublinear bipartiteness tester for bounded degree graphs. *Combinatorica*, 19(3):335–373, 1999.

[GR17] Tom Gur and Ron D Rothblum. A hierarchy theorem for interactive proofs of proximity. In *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[GR18] Tom Gur and Ron D Rothblum. Non-interactive proofs of proximity. *computational complexity*, 27(1):99–207, 2018.

[HA11] Mark Hillery and Erika Andersson. Quantum tests for the linearity and permutation invariance of boolean functions. *Physical Review A*, 84(6):062329, 2011.

[HHJ$^+$17] Jeongwan Haah, Aram W Harrow, Zhengfeng Ji, Xiaodi Wu, and Nengkun Yu. Sample-optimal tomography of quantum states. *IEEE Transactions on Information Theory*, 63(9):5628–5641, 2017.

[HHT93] Yenjo Han, Lane A. Hemaspaandra, and Thomas Thierauf. Threshold computation and cryptographic security. In *Algorithms and Computation*, pages 230–239, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

[HLM17] Aram W Harrow, Cedric Yen-Yu Lin, and Ashley Montanaro. Sequential measurements, disturbance and property testing. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1598–1611. SIAM, 2017.

[HM13] Aram W Harrow and Ashley Montanaro. Testing product states, quantum merlin-arthur games and tensor optimization. *Journal of the ACM (JACM)*, 60(1):1–43, 2013.

[KMS18] Subhash Khot, Dor Minzer, and Muli Safra. On monotonicity testing and boolean isoperimetric-type theorems. *SIAM Journal on Computing*, 47(6):2238–2276, 2018.

[KR15] Yael Tauman Kalai and Ron D Rothblum. Arguments of proximity. In *Annual Cryptology Conference*, pages 422–442. Springer, 2015.

[Lev85] L A Levin. One-way functions and pseudorandom generators. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, STOC '85, page 363–365, 1985.

[LKS$^+$19] Harry Levine, Alexander Keesling, Giulia Semeghini, Ahmed Omran, Tout T Wang, Sepehr Ebadi, Hannes Bernien, Markus Greiner, Vladan Vuletić, Hannes Pichler, et al. Parallel implementation of high-fidelity multiqubit gates with neutral atoms. *Physical review letters*, 123(17):170503, 2019.

[MdW13] Ashley Montanaro and Ronald de Wolf. A survey of quantum property testing. *arXiv:1310.2035*, 2013.

[MRR$^+$14] C Monroe, R Raussendorf, A Ruthven, KR Brown, P Maunz, L-M Duan, and J Kim. Large-scale modular quantum-computer architecture with atomic memory and photonic interconnects. *Physical Review A*, 89(2):022317, 2014.

[MW05] Chris Marriott and John Watrous. Quantum arthur–merlin games. *computational complexity*, 14(2):122–152, 2005.

[NV17] Anand Natarajan and Thomas Vidick. A quantum linearity test for robustly verifying entanglement. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1003–1015, 2017.

[OS03] Ryan O'Donnell and Rocco A. Servedio. New degree bounds for polynomial threshold functions. In Lawrence L. Larmore and Michel X. Goemans, editors, *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, June 9-11, 2003, San Diego, CA, USA*, pages 325–334. ACM, 2003.

[OW15] Ryan O'Donnell and John Wright. Quantum spectrum testing. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 529–538, 2015.

[OW16] Ryan O'Donnell and John Wright. Efficient quantum tomography. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 899–912, 2016.

[Pau02] Vern Paulsen. *Completely bounded maps and operator algebras.* Number 78. Cambridge University Press, 2002.

[Raz03] A A Razborov. Quantum communication complexity of symmetric predicates. *Izvestiya: Mathematics*, 67(1):145–159, Feb 2003.

[RKB18] Marc Olivier Renou, Jedrzej Kaniewski, and Nicolas Brunner. Self-testing entangled measurements in quantum networks. *Physical Review Letters*, 121(25):250507, 2018.

[RR20] Guy N Rothblum and Ron D Rothblum. Batch verification and proofs of proximity with polylog overhead. In *Theory of Cryptography Conference*, pages 108–138. Springer, 2020.

[RRR19] Omer Reingold, Guy N Rothblum, and Ron D Rothblum. Constant-round interactive proofs for delegating computation. *SIAM Journal on Computing*, (0):STOC16–255, 2019.

[RVW13] Guy N Rothblum, Salil Vadhan, and Avi Wigderson. Interactive proofs of proximity: delegating computation in sublinear time. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 793–802, 2013.

[ŠB19] Ivan Šupić and Joseph Bowles. Self-testing of quantum systems: a review. *arXiv:1904.10042*, 2019.

[ST19] Alexander A Sherstov and Justin Thaler. Vanishing-error approximate degree and QMA complexity. *arXiv:1909.07498*, 2019.

[Sti55] W Forrest Stinespring. Positive functions on C*-algebras. *Proceedings of the American Mathematical Society*, 6(2):211–216, 1955.

[TKV$^+$18] Armin Tavakoli, Jedrzej Kaniewski, Tamás Vértesi, Denis Rosset, and Nicolas Brunner. Self-testing quantum states and measurements in the prepare-and-measure scenario. *Physical Review A*, 98(6):062307, 2018.

[Wat09] John Watrous. Quantum computational complexity. In Robert A. Meyers, editor, *Encyclopedia of Complexity and Systems Science*, pages 7174–7201. Springer, 2009.

[WEH18] Stephanie Wehner, David Elkouss, and Ronald Hanson. Quantum internet: A vision for the road ahead. *Science*, 362(6412), 2018.

A A QCMAP lower bound for testing unitaries

The known $\mathcal{QMA}$ versus $\mathcal{QCMA}$ (oracle) separation of Aaronson and Kuperberg [AK07] is naturally cast as a testing problem, and thus yields a corresponding separation between $\mathcal{QMAP}$ and $\mathcal{QCMAP}$. More precisely, we consider the following property (of unitaries):

$$\Pi_{AK} = \left\{ U \in \mathbb{C}^{n\times n} : \begin{array}{l} UU^\dagger = U^\dagger U = I \text{ and } \exists\,|\psi\rangle \text{ such that } U|\psi\rangle = -|\psi\rangle \\ \text{and } U|\varphi\rangle = |\varphi\rangle \text{ when } \langle\varphi|\psi\rangle = 0 \end{array} \right\},$$

with the distance measure induced by the Hilbert-Schmidt norm: $d(U,V) = \mathrm{Tr}(U-V)^\dagger(U-V))$. Note that, for every $U \in \Pi_{AK}$,

$$d(U,I) = \sqrt{\sum_{i=1}^{n} \sigma_i(U-I)^2} = 2,$$

where $\sigma_i(A)$ is the i^{th} eigenvalue of A in nonincreasing order. Then $d(\Pi, I) = 2$, and we have

Theorem A.1. *For any proximity parameter $\varepsilon \leq 2$, we have $\Pi_{AK} \in \mathcal{QMAP}(\varepsilon, \log n, 1)$ and $\Pi_{AK} \notin \mathcal{QCMAP}(\varepsilon, p, q)$ when $p \cdot q = o(\sqrt{n})$.*

Proof. A $\mathcal{QMAP}$ algorithm with logarithmic proof length and query complexity 1 is as follows: given a $\log n$-qubit quantum state $|\psi\rangle$ as proof, apply the unitary to $|\psi\rangle$, accepting if and only if a phase flip is detected (by measuring and inspecting the outcome of the control qubit). Clearly, the eigenstate with eigenvalue -1 is a proof that causes the algorithm to accept with certainty if $U \in \Pi$, while no quantum state is accepted if $U = I$ (also with certainty).

For the lower bound, without loss of generality, we assume query access to a *controlled* unitary $U \in \Pi_{AK}$ or to the identity unitary. Note that, since the identity is 2-far from Π_{AK}, distinguishing between $U \in \Pi_{AK}$ and the identity is at least as hard as testing Π_{AK}. The lower bound proven in [AK07] can be expressed as follows: any QCMA algorithm that receives a proof of length p and oracle access to either $U \in \Pi_{AK}$ or the identity operator either accepts I or rejects some element of Π_{AK} with probability at least 1/2, if it makes $q = o(\sqrt{n/p})$ oracle queries. Since $d(\Pi, I) = 2 \geq \varepsilon$, the result follows. □

B Interaction versus quantum proofs

In this section we compare the power of classical interactive proofs of proximity (IPPs) and non-interactive quantum proofs of proximity (QMAPs), and show that the rather well studied problem of *permutation testing* admits an efficient IPP but no efficient QMAP. In fact, for permutation testing, even an Arthur-Merlin Proof of Proximity is sufficient, as shown in [GLR18]. Informally, an Arthur-Merlin Proof of Proximity (AMP) is a proof system with one round of communication where the verifier sends the first message.

Let Π_P be property (of *functions* $f : [n] \to [n]$) defined as

$$\Pi_P = \{f : f \text{ is a bijection}\} \ ,$$

i.e., Π_P is the set of all permutations. We note that in an oracle query to f, an algorithm sends $x \in [n]$ and receives (the $\log n$-bit string) $f(x)$; accordingly, a quantum query maps $|x\rangle |y\rangle \mapsto |x\rangle |y + f(x)\rangle$ (both $\log n$-qubit states). Moreover, distance is measured in terms of the fraction of inputs where functions disagree (rather than with respect to their representations as bit strings), i.e., f and g are ε-far when $|\{x \in [n] : f(x) \neq g(x)\}|/n \geq \varepsilon$.

The separation follows immediately from the two following theorems.

Theorem B.1 ([GLR18, Lemma 4.2])**.** *For every with $\varepsilon > 0$, There exists an IPP for ε-testing Π_P with query complexity $O(1/\varepsilon)$ and communication complexity $O(\log n/\varepsilon)$, that communicates two messages: the first is sent from verifier to prover and the second from prover to verifier. Therefore,*

$$\Pi_P \in \mathcal{AMP}(\varepsilon, O(\log n/\varepsilon), O(1/\varepsilon), 2) \ .$$

Since $\mathcal{AMP}(\varepsilon, c, q, r) \subseteq \mathcal{IPP}(\varepsilon, c, q, r+1)$ (by initiating the protocol with a "dummy" message by the prover), the following corollary is immediate.

Corollary 10. $\Pi_P \in \mathcal{IPP}(\varepsilon, O(\log n/\varepsilon), O(1/\varepsilon), 3)$.

Having established that Π_P admits an efficient IPP, we must now show it is *not* efficiently testable by a QMAP:

Lemma 9 ([ST19, Theorem 1.2])**.** *Any QMAP protocol for testing* Π *with respect to proximity parameter* $\varepsilon = \Omega(1)$*, using a proof of length* p *and making* q *queries, satisfies* $p \cdot q^3 = \Omega(n)$*; i.e.,*

$$\Pi_P \notin \mathcal{QMAP}(\varepsilon, p, q) \text{ when } p \cdot q^3 = o(n) \ .$$

Note that this implies that either p or q must be $\Omega(n^{1/4})$. Finally, Theorem B.1 and Lemma 9 together imply the main result of this section.

Theorem B.2. *Let* $\Pi_P \subset \{f : [n] \to [n]\}$ *be the set of bijective functions from* $[n]$ *to* $[n]$*, with the distance between functions* f *and* g *defined as* $|\{x \in [n] : f(x) \neq g(x)\}|/n$*. Then, for any* $\varepsilon = \Omega(1)$*,*

$$\begin{aligned} &\Pi_P \in \mathcal{IPP}(\varepsilon, O(\log n), O(1), 3) \text{ and} \\ &\Pi_P \notin \mathcal{QMAP}(\varepsilon, p, q) \text{ when } p \cdot q^3 = o(n) \ . \end{aligned}$$

In particular,

$$\mathcal{IPP}(\varepsilon, O(\log n), O(1), 3) \not\subseteq \mathcal{QMAP}(\varepsilon, o(n^{1/4}), o(n^{1/4})) \ .$$