Worst-Case to Average-Case Reductions via Additive Combinatorics

Vahid R. Asadi* Alexander Golovnev† Tom Gur‡ Igor Shinkar§

Abstract

We present a new framework for designing worst-case to average-case reductions. For a large class of problems, it provides an explicit transformation of algorithms running in time T that are only correct on a small (subconstant) fraction of their inputs into algorithms running in time $\widetilde{O}(T)$ that are correct *on all* inputs.

Using our framework, we obtain such efficient worst-case to average-case reductions for fundamental problems in a variety of computational models; namely, algorithms for matrix multiplication, streaming algorithms for the online matrix-vector multiplication problem, and static data structures for all linear problems as well as for the multivariate polynomial evaluation problem.

Our techniques crucially rely on additive combinatorics. In particular, we show a local correction lemma that relies on a new probabilistic version of the quasi-polynomial Bogolyubov-Ruzsa lemma.

*University of Waterloo. Email: vrasadi@uwaterloo.ca.

†Georgetown University. Email: alexgolovnev@gmail.com.

‡University of Warwick. Email: tom.gur@warwick.ac.uk. Tom Gur is supported by the UKRI Future Leaders Fellowship MR/S031545/1.

§Simon Fraser University. Email: ishinkar@sfu.ca.

Contents

1 Introduction

Worst-case to average-case reductions provide a method for transforming algorithms that can only solve a problem for a fraction of the inputs into algorithms that can solve the problem *for all* inputs.

For instance, consider one of the most fundamental algorithmic problems: matrix multiplication. Suppose we have an average-case algorithm ALG that can correctly compute the product $A \cdot B$ on an α-fraction of matrices $A, B \in \mathbb{F}^{n \times n}$; that is, $\Pr[\mathsf{ALG}(A, B) = A \cdot B] \geq \alpha$. Is it possible to use ALG to obtain an algorithm that computes $A \cdot B$ *for all* input matrices? A worst-case to average-case reduction will give a positive answer to this question, boosting the *success rate* α to 1, without incurring significant overhead. Of course, the same question can be asked with respect to any other computational problem.

In this paper, we study such reductions for average-case algorithms where the success rate α could be very small, such as in the %1 regime, and even when α tends to zero rapidly (i.e., for algorithms that are only correct on a vanishing fraction of their inputs). There are two natural perspectives in which we can view such reductions. On the one hand, they can provide a proof that a problem retains its hardness even in the average case. On the other hand, they provide a paradigm for designing worst-case algorithms, by first constructing algorithms that are only required to succeed on a small fraction of their inputs, and then using the reduction to obtain algorithms that are correct on all inputs.

Background and context. The study of the average-case complexity originates in the work of Levin [Lev86]. A long line of works established various barriers to designing worst-case to average-case reductions for **NP**-complete problems (see, e.g., [Imp11] and references therein). We refer the reader to the classical surveys by Impagliazzo [Imp95], and Bogdanov and Trevisan [BT06] on this topic.

On the positive side, Lipton [Lip91] proved that the matrix permanent problem admits a polynomial-time worst-case to average-case reduction. Ajtai [Ajt96] designed worst-case to average-case reductions for certain lattice problems, which led to constructions of efficient cryptographic primitives from worst-case assumptions [AD97, Reg04]. Other number-theoretic problems in cryptography have been long known to admit such reductions due to random self-reducibility: the discrete logarithm problem, the RSA problem, and the quadratic residuosity problem (see, e.g., [Sho09]). For the matrix multiplication problem, there is a weak reduction that requires the average-case algorithm to succeed with very high probability 3/4 (see Section 2.1). There are also known worst-case to average-case reductions for many problems that are not thought to be in **NP** [FF93, BFNW93, STV01].

Recently, the study of fine-grained complexity [Vas18] of algorithmic problems sparked interest in designing *efficient* worst-case to average-case reductions for such problems as orthogonal vectors, 3SUM, online matrix-vector multiplication, k-clique, and others. Such reductions are motivated by fine-grained cryptographic applications. A large body of work is devoted to establishing fine-grained worst-case to average-case reductions for the k-clique problem, orthogonal vectors, 3SUM, and various algebraic problems, as well as building certain cryptographic primitives from them [BRSV17, BRSV18, GR18, LLV19, BABB19, DLV20]. Since there are no known constructions of one-way functions and public-key cryptography from well-established fine-grained assumptions, the question of constructing efficient worst-case to average-case reductions for other fine-grained problems still attracts much attention.

1.1 Our contribution

We design a framework for showing explicit worst-case to average-case reductions, and we use it to obtain reductions for fundamental problems in a variety of computational models. Informally, we show that if a problem has an algorithm that runs in time T and succeeds on α-fraction of its inputs (even for sub-constant success rate α), then there exists a worst-case algorithm for this problem, which runs in time $\widetilde{O}(T)$. We design such reductions for the matrix multiplication problem in the setting of algorithms, for the online matrix-vector multiplication problem in the streaming setting, for *all* linear problems in the setting of static data structures, and for the problem of multivariate polynomial evaluation. We describe these results in detail below.

1.1.1 Algorithms for matrix multiplication

Recall that in the matrix multiplication problem, the goal is simply to compute the product of two given matrices $A, B \in \mathbb{F}^{n\times n}$. A long line of research, culminating in the work of Alman and Vassilevska Williams [AV21], led to matrix multiplication algorithms performing $O(n^{2.37286})$ operations. We present a worst-case to average-case reduction for the matrix multiplication problem over prime fields. Namely, we show that if there exists a (randomized) algorithm that, given two matrices $A, B \in \mathbb{F}^{n\times n}$, runs in time $T(n)$ and correctly computes their product for a small fraction of all possible inputs, then there exists a (randomized) algorithm that runs in $\widetilde{O}(T(n))$ time and outputs the correct answer for all inputs. Formally, we have the following theorem.

Theorem 1. *Let $\mathbb{F} = \mathbb{F}_p$ be a prime field, $n \in \mathbb{N}$, and $\alpha := \alpha(n) \in (0, 1]$. Suppose that there exists an algorithm ALG that, on input two matrices $A, B \in \mathbb{F}^{n\times n}$ runs in time $T(n)$ and satisfies*

$$\Pr[\mathsf{ALG}(A, B) = A \cdot B] \geq \alpha\,,$$

where the probability is taken over the random inputs $A, B \in \mathbb{F}^{n\times n}$ and the randomness of ALG.

- *If $|\mathbb{F}| \leq 2/\alpha$, then there exists a randomized algorithm ALG' that for* every *input $A, B \in \mathbb{F}^{n\times n}$ and $\delta > 0$, runs in time $\frac{\exp(O(\log^5(1/\alpha)))}{\delta} \cdot T(n)$ and outputs AB with probability at least $1 - \delta$.*
- *If $|\mathbb{F}| \geq 2/\alpha$, then there exists a randomized algorithm ALG' that for* every *input $A, B \in \mathbb{F}^{n\times n}$ and $\delta > 0$, runs in time $O(\frac{1}{\delta\cdot\alpha^4} \cdot T(n))$ and outputs AB with probability at least $1 - \delta$.*

For example, if we have an algorithm that succeeds on α fraction of the inputs for $\alpha > \exp(-\sqrt[6]{\log(n)})$ in time $T(n) = n^c$, then we get an algorithm that works for all inputs and runs in time $n^{c+o(1)}$. In particular, if we have an $n^{2+o(1)}$ algorithm that succeeds on $\alpha > \exp(-\sqrt[6]{\log(n)})$ fraction of the inputs, then there is a worst case algorithm with running time $n^{2+o(1)}$.

1.1.2 Data structures for all linear problems

The class of linear problems plays a central role throughout computer science and mathematics, yielding a myriad of applications both in theory and practice. Our next contribution gives worst-case to average-case reductions for static data structures for *all linear problems*. Recall that a linear problem L_A over a field $\mathbb{F}$ is defined by a matrix $A \in \mathbb{F}^{m\times n}$.[1] An input to the problem is a vector $v \in \mathbb{F}^n$, which is preprocessed into s memory cells. Then, given a query $i \in [m]$, the goal is to

[1] Formally, L_A is defined by an infinite sequence of matrices $(A_n)_{n\geq 1}$, where $A_n \in \mathbb{F}^{m\times n}$ for $m = m(n)$.

output $\langle A_i, v\rangle$, where A_i is the i'th row of A, by probing at most t of the memory cells, where t is called the query time.

Note that the trivial solutions for data structure problems are to either: (i) store only $s = n$ memory cells containing the input v, and for each query $i \in [m]$, read v entirely and compute the answer in query time $t = n$; or (ii) use $s = m$ memory cells, where the i'th cell contains the answer to the query $i \in [m]$, thus allowing for query time $t = 1$. In a typical application, the number of queries $m = \text{poly}(n) \gg n$, and a data structure is efficient if it uses space $s = \widetilde{O}(n)$ (or $s \ll m$) and has query time $t = \text{poly}(\log(n))$ (or $t = n^{\varepsilon}$ for a small constant $\varepsilon > 0$). Note that the two trivial solutions do not lead to such efficient data structures for $m \gg n$.

We consider randomized data structures, where both the preprocessing stage and the query stage use randomness, and are expected to output the correct answer with high probability (over the randomness of both stages). In average-case randomized data structures, the success rate of the algorithm is taken over both the inner randomness and the random input, whereas in worst-case randomized data structure, the success rate is taken only over the inner randomness of the algorithm (i.e., the algorithm succeeds with high probability *on all* inputs).

We present a worst-case to average-case reduction showing that if there exists a data structure DS that uses s memory cells, has query time t, and success rate such that for a small fraction of inputs the data structure answers all queries correctly, then there exists another data structure DS' that uses $4s$ memory cells, has query time $4t$, and success rate such that *for all inputs* the data structure answers all queries correctly with high probability.

Theorem 2. *Let $\mathbb{F} = \mathbb{F}_p$ be a prime field, $\alpha := \alpha(n) \in (0, 1]$, $n, m \in \mathbb{N}$, and a matrix $A \in \mathbb{F}^{m \times n}$. Denote by L_A the linear problem of outputting $\langle A_i, x\rangle$ on input $x \in \mathbb{F}^n$ and query $i \in [m]$. Suppose that*

$$L_A \in \mathsf{DS}\left[\begin{array}{rl} \textit{preprocessing time:} & p \\ \textit{memory used:} & s \\ \textit{query time:} & t \\ \textit{success rate:} & \Pr_{x\in\mathbb{F}^n}[\mathsf{DS}_x(i) = \langle A_i, x\rangle\, \forall i \in [m]] \geq \alpha \end{array}\right].$$

Then for every $\delta > 0$,

$$L_A \in \mathsf{DS}\left[\begin{array}{rl} \textit{preprocessing time:} & p + \exp(\log^4(1/\alpha)) \cdot \text{poly}\log(1/\delta) \cdot \text{poly}(n) \\ \textit{memory used:} & 4s + O(\log^4(1/\alpha)\log(n)) \\ \textit{query time:} & 4t + O(\log^4(1/\alpha)\log(n)) \\ \textit{success rate:} & \forall x \in \mathbb{F}^n \;\; \Pr[\mathsf{DS}'_x(i) = \langle A_i, x\rangle\, \forall i \in [m]] \geq 1 - \delta \end{array}\right].$$

We stress that in the average-case data structure we start with, the probability is taken over a random input (as well as the inner randomness of the algorithm), whereas in the worst-case data structure that we obtain, with high probably the algorithm is successful *on all* inputs.

The reduction above shows that for any linear problem L_A, if a data structure succeeds on an arbitrary small constant $\alpha > 0$ fraction of the inputs, then we can obtain a data structure that succeeds on all inputs with parameters that essentially differ only by a constant multiplicative factor, and the query complexity t translates into query complexity $4t + O(\log(n))$.

We note that the $O(\log^4(1/\alpha)\log(n))$ overhead in the space complexity of the constructed data structure is caused by storing $O(\log^4(1/\alpha))$ numbers from $[n]$. In particular, if the word size of the data structure is $w \geq \log(n)$, then the space complexity of the resulting data structure is $4s + O(\log^4(1/\alpha))$. Similarly, in this case the query complexity of the resulting data structure is $4t + O(\log^4(1/\alpha))$.

Note that for any non-trivial data structure problem, a data structure must use at least $\Omega(n)$ memory cells (only to store a representation of the input). Therefore, even for α as small as $\alpha = 2^{-n^\eta}$ for a small constant $\eta > 0$, the overhead in the space complexity is negligible. For typical query times of data structures, such as $t = \text{poly}(\log(n))$ and $t = n^\varepsilon$, the overhead in the query time is negligible even for $\alpha = 1/\text{poly}(n)$ and $\alpha = 2^{-n^\eta}$, respectively.

1.1.3 Online matrix-vector multiplication

Next we turn to the core data structure problem in fine-grained complexity, the online matrix-vector multiplication problem (OMV). In the data structure variant of this streaming problem, one needs to preprocess a matrix $M \in \mathbb{F}^{n\times n}$, such that given a query vector $v \in \mathbb{F}^n$, one can quickly compute Mv. The study of OMV (over the Boolean semiring) and its applications to fine-grained complexity originates from [HKNS15], and [LW17, CKL18] give surprising upper bounds for the problem. Over finite fields, [FHM01, CGL15] give lower bounds for OMV, and [CKLM18] proves lower bounds for a related vector-matrix-vector multiplication problem. We prove an efficient worst-case to average-case reduction for OMV over prime fields. A concurrent and independent work [HLS21] studies worst-case to average-case reductions for OMV over the Boolean semiring and their applications.

Note that OMV is, in fact, *not* a linear problem, because for a query v the output is not a single field element, but rather a vector $Mv \in \mathbb{F}^n$. Moreover, the average case condition only guarantees success with probability taken over *both* the matrix M as well as the vector v. Nevertheless, we can exploit the fact that each coordinate of the correct output is a linear function in the entries of M, and extend our techniques to the more involved setting of OMV.

Theorem 3. *Let $\mathbb{F} = \mathbb{F}_p$ be a prime field, $n \in \mathbb{N}$, and $\alpha := \alpha(n) \in (0, 1]$. Consider the matrix-vector multiplication problem $OMV_\mathbb{F}$ for dimension n, and suppose that for some $\alpha > 0$ it holds that*

$$OMV_\mathbb{F} \in \mathsf{DS}\left[\begin{array}{rl} \textit{preprocessing time:} & p \\ \textit{memory used:} & s \\ \textit{query time:} & t \\ \textit{success rate:} & \Pr_{M,v}[\mathsf{DS}_M(v) = Mv] \geq \alpha \end{array}\right].$$

Then for every $\delta > 0$,

$$OMV_\mathbb{F} \in \mathsf{DS}\left[\begin{array}{rl} \textit{preprocessing time:} & 4p + \exp(\log^4(1/\alpha)) \cdot \text{poly}\log(1/\delta) \cdot \text{poly}(n) \\ \textit{memory used:} & 4s + O(\log^4(1/\alpha)n) + O(n^2) \\ \textit{query time:} & (4t + n) \cdot \text{poly}(1/\alpha) \cdot \text{poly}\log(1/\delta) \\ \textit{success rate:} & \forall M, v : \Pr[\mathsf{DS}_M(v) = Mv] \geq 1 - \delta \end{array}\right].$$

We stress that in the assumed data structure, the success rate asserts that for *a random input M and query v*, the data structure produces the correct answer with (an arbitrary small) probability $\alpha > 0$, where the probability is over (i) the random input M (ii) random query v (iii) and the randomness of the preprocessing and the query phases of the data structure. On the other hand, the conclusion holds for worst case inputs and queries. That is, *for every* input M and query v, the obtained data structure produces the correct answer with high probability, where the probability is only over the randomness used in the preprocessing stage and the query phase of the data structure (i.e., with high probability we can compute *all* of the inputs).

To understand the parameters of the reduction, note that in the the OMV problem with $n \times n$ matrices, the preprocessing must be at least n^2, as this is the size of the input matrix, and the query time must be at least n, as information-theoretically we need to output n field elements. Our

worst-case to average-case reduction is essentially optimal in these parameters for a constant α, as a weak data structure that uses s memory cells and query time t is translated into a data structure that works for all inputs and all queries using space $4s + O(n^2)$ and query time $4t + O(n) = O(t)$. In fact, even for α as small as $\alpha = 1/n^{o(1)}$, the space complexity is increased by at most $O(n^2)$, and the query time is multiplied by at most $n^{o(1)}$.

1.1.4 Worst-case to weak-average-case reductions

In the following, we discuss how to obtain worst-case algorithms starting from a very weak, but natural, notion of average-case reductions that we discuss next.

Recall that in the standard definition of average-case data structures, the algorithm preprocesses its input and is then required to correctly answer all queries for an α-fraction of all possible inputs. However, in many cases (such as in the online matrix-vector multiplication problem), we only have an average-case guarantee on both inputs and queries. In this setting, we should first ask what is a natural notion of an average-case condition.

A strong requirement for an average-case algorithm in this case is to correctly answer *all queries* for at least α-fraction of the inputs. However, it is desirable to only require the algorithm to correctly answer on an *average input and query*. That is, a *weak average-case* data structure for computing a function $f\colon \mathbb{F}^n \times [m] \to \mathbb{F}$ with success rate $\alpha > 0$ receives an input $x \in \mathbb{F}^n$, which is preprocessed into s memory cells. Then, given a query $i \in [m]$, the data structure $\mathsf{DS}_x(i)$ outputs $y \in \mathbb{F}^{n'}$ such that $\Pr_{x\in\mathbb{F}^n, i\in[m]}[\mathsf{DS}_x(i) = f(x,i)] \geq \alpha$.

The challenge in this setting is that the errors may be distributed between both the inputs and the queries. On one extreme, the error is concentrated on selected inputs, and then the data structure computes *all queries* correctly for α-fraction of the inputs. On the other extreme, the error is spread over all inputs, and then the data structure may only answer α-fraction of the queries on any inputs. Of course, the error could be distributed anywhere in between these extremes.

While we showed that every linear problem has an efficient worst-case to average-case reduction, in Section 6.3 we show that not all linear (and non-linear) problems admit a worst-case to *weak*-average-case reductions. Nevertheless, we overcome this limitation for certain problems of interest.

One of the most-studied problems in static data structures is the polynomial evaluation problem [KU08, Lar12, DKKS21]. Here, one needs to preprocess a degree-d polynomial $q\colon \mathbb{F}^m \to \mathbb{F}$ into s memory cells, and then for a query $x \in \mathbb{F}^m$, quickly compute $q(x)$. We study the problem of evaluating a low degree polynomial in the regime where the average-case data structure might only succeed on a small α fraction of the queries (outside of the unique decoding regime, see discussion below). We show that we can use such an average-case data structure to obtain a worst-case data structure that can compute q on any $x \in \mathbb{F}^m$.

Theorem 4. *Let $\mathbb{F} = \mathbb{F}_p$ be a prime field, $\alpha := \alpha(n) \in (0,1]$, and let $m, d \in \mathbb{N}$ be parameters. Consider the problem $\mathsf{RM}_{\mathbb{F},\mathsf{m},\mathsf{d}}$ of evaluating polynomials of the form $q\colon \mathbb{F}^m \to \mathbb{F}$ of total degree d (i.e., the problem of evaluating the Reed-Muller encoding of block length $n = \binom{m+d}{d}$).*

Suppose that

$$\mathsf{RM}_{\mathbb{F},\mathsf{m},\mathsf{d}} \in \mathsf{DS}\left[\begin{array}{rl} \textit{preprocessing time:} & p \\ \textit{memory used:} & s \\ \textit{query time:} & t \\ \textit{success rate:} & \Pr_{q,x}[\mathsf{DS}_q(x)] \geq \alpha \end{array}\right].$$

Then

$$\mathsf{RM}_{\mathbb{F},\mathsf{m},\mathsf{d}} \in \mathsf{DS}\left[\begin{array}{rl} \textit{preprocessing time:} & p + \exp(\log^4(1/\alpha)) \cdot \mathrm{poly}(n) \\ \textit{memory used:} & 4s + O(\log^4(1/\alpha)\log(n)) \\ \textit{query time:} & O(|\mathbb{F}|^2 \cdot t + |\mathbb{F}|\log^4(1/\alpha) + |\mathbb{F}|\log(n)) \\ \textit{success rate:} & \forall q, x : \Pr[\mathsf{DS}_q(x) = q(x)] > 1 - O\left(\sqrt{\frac{d}{|\mathbb{F}|}}\right) \end{array}\right].$$

Here, similarly to Theorem 3, the assumed data structure succeeds only for a small fraction of inputs and queries, while in the conclusion the data structure succeeds with high probability on *every input* and *every query*.

As for the effect of the reduction on the parameters, we see that for any $\alpha > 1/\mathrm{poly}(n)$ the preprocessing time changes only by an additive $\mathrm{poly}(n)$, the space complexity changes from s to $4s + \mathrm{poly}(\log(n))$, and the query time changes from t to $O(|\mathbb{F}|^2 \cdot t + |\mathbb{F}| \cdot \mathrm{poly}\log(n))$. In the data structure setting, the number of queries is usually polynomial in input length. Thus, in a typical setting of parameters for $\mathsf{RM}_{\mathbb{F},\mathsf{m},\mathsf{d}}$, the field size is $|\mathbb{F}| = \mathrm{poly}(\log n)$, and, therefore, the blow-up of $|\mathbb{F}|^2$ is not critical.

A coding-theoretic perspective. For small values of average-case rate $\alpha > 0$, the polynomial evaluation problem can be cast as *list decoding with preprocessing*, by viewing the outputs of the query phase of the data structure as a function $h\colon \mathbb{F}^m \to \mathbb{F}$ that agrees with the input polynomial $q\colon \mathbb{F}^m \to \mathbb{F}$ on some small fraction of the queries, and the goal is to recover q from h.

Indeed, note that for a small $\alpha > 0$, if a function $h\colon \mathbb{F}^m \to \mathbb{F}$ agrees with some unknown low-degree polynomial q on α fraction of the inputs, then there are potentially $O(1/\alpha)$ possible low-degree polynomials that are equally close to h. Hence, without preprocessing it is impossible to recover the original polynomial q. However, in the data structure settings, we can use the preprocessing to obtain an auxiliary structural information that would later allow us to transition from the list decoding regime to the unique decoding regime, and in turn, compute the values of the correct polynomial q with high probability (see more details in Section 2.4).

1.2 Open problems

Our work leaves many natural open problems, such as obtaining reductions for various natural problems in other computational models (e.g., communication complexity, property testing, PAC learning, and beyond). However, for brevity, we would like to focus on and highlight one direction that we find particularly promising.

In Theorem 2, we design worst-case to average-case reductions for linear problems in the setting of static data structures. An immediate and alluring question is whether our local correction via additive combinatorics framework can also be used to show worst-case to average-case reductions for all linear problems for both circuits and uniform algorithms. We observe that using similar techniques as in Theorem 2, our framework can be used to show that given an efficient average-case circuit or uniform algorithm and an efficient *verifier* for the problem, one can indeed design an explicit efficient worst-case circuit or uniform algorithm. A natural open problem here is to eliminate the assumption about the verifier and answer the aforementioned question to the affirmative.

Acknowledgments

We are grateful to Tom Sanders for providing a sketch of the proof of the probabilistic version of the quasi-polynomial Bogolyubov-Ruzsa lemma. We would also like to thank Shachar Lovett and Tom Sanders for discussions regarding the quasi-polynomial Bogolyubov-Ruzsa lemma.

2 Technical overview

We provide an overview of the main ideas and techniques that we use to obtain our results. For concreteness, we illustrate our techniques by first focusing on the matrix multiplication problem.

We start in Section 2.1, where we explain the challenge and discuss why the naive approach fails. In Section 2.2 we present the technical components that lie at the heart of this work: *local correction lemmas via additive combinatorics*. Equipped with these technical tools, in Section 2.3 we present the main ideas in our worst-case to average-case reduction for matrix multiplication. Finally, in Section 2.4 we briefly discuss how to obtain the rest of our main results.

2.1 The challenge: low-agreement regime

Recall that in the matrix multiplication problem we are given two matrices $A, B \in \mathbb{F}^n$, and the goal is to compute their matrix product $A \cdot B$. For simplicity of the exposition, unless specified otherwise, in this overview we restrict our attention to the field $\mathbb{F}_2$, and to constant values of the success rate parameter $\alpha > 0$ of average-case algorithms.

We would like to show that if there is an *average-case* algorithm ALG that can compute matrix multiplication for an α-fraction of all pairs of matrices $A, B \in \mathbb{F}^n$ in time $T(n)$, then there is a *worst-case* randomized algorithm ALG' that runs in time $O(T(n))$ and computes $A \cdot B$ with high probability for *every* pair of matrices A and B.

We start with the elementary case where the average-case guarantee is in the *high-agreement regime*, i.e., where the algorithm succeeds on, say, 99% of the inputs; that is,

$$\Pr_{A,B \in \mathbb{F}^{n \times n}} [\mathsf{ALG}(A, B) = A \cdot B] \geq \alpha \,, \tag{1}$$

for $\alpha = 0.99$. In this case, a folklore local correction procedure (see, e.g., [BLR90]) will yield a worst-case algorithm that succeeds with high probability on all inputs. We next describe this procedure.

Given an *average-case* algorithm ALG satisfying Eq. (1) with $\alpha = 0.99$, consider the worst-case algorithm ALG' that receives any two matrices $A, B \in \mathbb{F}^{n \times n}$ and first samples uniformly at random two matrices $R, S \in \mathbb{F}^{n \times n}$. Next, writing $A = R + (A - R)$ and $B = S + (B - S)$, the algorithm ALG' computes

$$M = \mathsf{ALG}(R, S) + \mathsf{ALG}(A - R, S) + \mathsf{ALG}(R, B - S) + \mathsf{ALG}(A - R, B - S) \,. \tag{2}$$

Denote by X the set of matrix pairs (A, B) for which $\mathsf{ALG}(A, B) = A \cdot B$, and recall that by Eq. (1) the density of X is 0.99. Note that: (a) the matrices R, $A - R$, S, and $B - S$ are uniformly distributed, and (b) if the pairs (R, S), $(A - R, S)$, $(R, B - S)$, and $(A - R, B - S)$ are in the set X, then by Eq. (2) we have $M = A \cdot B$, and the algorithm ALG' computes the multiplication correctly. Hence, by a union bound we have $\Pr[M = AB] \geq 1 - 4 \cdot 0.01 > 0.9$ *for all* matrices $A, B \in \mathbb{F}^{n \times n}$.

Of course, the error probability can be further reduced by repeating the procedure and ruling by majority.

Unfortunately, this argument breaks when the average-case guarantee is weaker; namely, in the *low-agreement regime*, where the algorithm succeeds on, say, only 1% of the inputs. Here, when trying to self-correct as above, the vast majority of random choices would lead to a wrong output, and so at a first glace, the self-correction approach may seem completely hopeless.[2]

Nevertheless, using more involved tools from additive combinatorics such as a probabilistic version of the quasi-polynomial Bogolyubov-Ruzsa lemma that we show, as well as tools such as small-biased sample spaces and the Goldreich-Levin algorithm, we can construct different local correction procedures that work in the *low-agreement regime*. We proceed to describe our framework for local correction using the aforementioned tools.

2.2 Local correction via additive combinatorics

Additive combinatorics studies approximate notions of algebraic structures via the perspective of combinatorics, number theory, harmonic analysis and ergodic theory. Most importantly for us, it provides tools for transitioning between algebraic and combinatorial notions of approximate subgroups with only a small loss in the underlying parameters (see surveys [Lov15, Lov17]).

The starting point of our approach for local correction is a fundamental result in additive combinatorics, known as *Bogolyubov's lemma*, which shows that the 4-ary sumset of any dense set in $\mathbb{F}_2^n$ contains a large linear subspace. More accurately, recall that the sumset of a set X is defined as $X + X = \{x_1 + x_2 : x_1, x_2 \in X\}$, and similarly $4X = \{x_1 + x_2 + x_3 + x_4 : x_1, x_2, x_3, x_4 \in X\}$. These quantities can be thought of as quantifying a combinatorial analogue of an approximate subgroup. Bogolyubov's lemma states that for any subset $X \subseteq \mathbb{F}_2^n$ of density $|X|/2^n \geq \alpha$, there exists a subspace $V \subseteq 4X$ of dimension at least $n - \alpha^{-2}$.

We will show that statements of the above form can be used towards obtaining a far stronger local correction paradigm than the one outlined in Section 2.1. To see the initial intuition, consider an average-case algorithm that is guaranteed to correctly compute α-fraction of the inputs, and denote by X the set of these correctly computed inputs. Then $|X|/2^n \geq \alpha$, and Bogolyubov's lemma shows that there exists a large subspace V such that every $v \in V$ can be expressed as a sum of four elements in X, each of which can be computed correctly by the average-case algorithm.

The approach above suggests a paradigm for local correction, however, there are several non-trivial problems in implementing this idea. For starters, how could we handle inputs that lay *outside of the subspace* V? To name a few others: how can we amplify the success probability in the low-agreement regime? How do we algorithmically obtain the decomposition? Can we handle finite fields beyond $\mathbb{F}_2^n$? How do we handle average-case where the success rate α is sub-constant?

Indeed, for our worst-case to average-case reductions, we will need local correction lemmas with stronger structural properties than those admitted by Bogolyubov's lemma, as well as new ideas for each one of the settings. In the following, we discuss the main hurdles for the foregoing approach and the tools that are needed to overcome them, leading to our main technical tool, which is a probabilistic version of the quasi-polynomial Bogolyubov-Ruzsa lemma that we obtain. Then, we

[2]Indeed, consider the counterexample where the average-case algorithm $\mathsf{ALG}(A, B)$ outputs $A \cdot B$ in case the first element of A is 0 and returns the zero matrix in case the first element of A is 1. Note that in this case $\Pr_{A,B \in \mathbb{F}^{n \times n}}[\mathsf{ALG}(A, B) = A \cdot B] \geq 1/2$, yet no decomposition of $A = \sum_i A_i$ and $B = \sum_i B_i$ as described above could self-correct matrix multiplication where the first element of A is 1. Indeed, any such composition would have an A_i with the first element 1, where $\mathsf{ALG}(A_i, B_j)$ fails.

present our framework for local correction using these techniques. Finally, in Sections 2.3 and 2.4 we show the additional ideas that are necessary for applying the local correction lemmas in the settings of matrix multiplication, online matrix-vector multiplication, and data structures.

A probabilistic Bogolyubov lemma. An immediate problem with the aforementioned local correction scheme is that while Bogolyubov's lemma asserts that *there exists* a decomposition of each input into a sum of four elements in X, it does not tell us how to obtain this decomposition.

Toward this end, we further show that each vector $v \in V$ has many "representations" as a sum of four elements from X. This way, for any $v \in V$ we can efficiently sample a representation $v = x_1 + x_2 + x_3 + x_4$, where each $x_i \in X$. More accurately, let $X \subseteq \mathbb{F}_2^n$ be a set of density α, let $R = \{r \in \mathbb{F}^n \setminus \{0\} : \left|\hat{1}_X(r)\right| \geq \alpha^{3/2}\}$, and let $V = \{v \in \mathbb{F}^n : \langle v, r\rangle = 0 \ \forall r \in R\}$ be a linear subspace defined by R. Then $|R| \leq 1/\alpha^2$ and for all $v \in V$ it holds that

$$\Pr_{x_1,x_2,x_3}[x_1, x_2, x_3, v - x_1 - x_2 - x_3 \in X] \geq \alpha^5 \ .$$

Sparse-shift subspace decomposition. The probabilistic Bogolyubov lemma allows us to locally correct inputs inside the subspace $V \subseteq 4X$. However, we need to be able to handle any vector in the field. Towards that end, we show an algebraic lemma that allows us to decompose each element of the field into a sum of an element v in the subspace V and a *sparse* shift-vector s. More accurately, let $R \subseteq \mathbb{F}^n \setminus \{\vec{0}\}$ and $V = \{v \in \mathbb{F}^n : \langle v, r\rangle = 0 \ \forall r \in R\}$. We show that there exists a collection of $t \leq |R|$ vectors $B = \{b_1, \ldots, b_t\}$, $b_i \in \mathbb{F}^n$ and indices $k_1, \ldots, k_t \in [n]$ such that $\mathsf{span}(B) = \mathsf{span}(R)$ and every vector $y \in \mathbb{F}^n$ can be written as $y = v + s$, where $v \in V$ and $s = \sum_{j=1}^{t} c_j \cdot \vec{e}_{k_j}$ for $c_j = \langle y, b_j\rangle$ and $\vec{e}_{k_j}$ is a unit vector.

We stress that the sparsity of the decomposition is essential to our applications, as we cannot locally correct the shift part of the decomposition, and instead we need to compute it explicitly. We remark that for matrix multiplication we can obtain a stronger guarantee by dealing with matrices outside of the subspace V via a low-rank random matrix shifts (see Section 2.3).

Subspace computation via the Goldreich-Levin lemma. In order to perform local correction using additive combinatorics machinery as above while maintaining computational efficiency, we need to be able to compute the aforementioned basis $b_1, \ldots, b_t \in \mathbb{F}^n$ and indices $k_1, \ldots, k_t \in [n]$ efficiently. We note that, in essence, this problem reduces to learning the heavy Fourier coefficients of the set X. Thus, using ideas from [BRTW14] and an extension of the Goldreich-Levin algorithm to arbitrary finite fields, we can perform the latter in a computationally efficient way.

Probabilistic quasi-polynomial Bogolyubov-Ruzsa lemma. The main weakness of Bogolyubov's lemma is that the co-dimension of the subspace that it admits is polynomial in $1/\alpha$, where α is the success rate of the average-case algorithm. While this dependency on α allows us to locally correct in the 1% agreement regime, it becomes degenerate when α tends to 0 rapidly.

A natural first step towards overcoming this barrier is to use a seminal result due to Sanders [San12], known as the *quasi-polynomial Bogolyubov-Ruzsa lemma*, which shows the existence of a subspace whose co-dimension's dependency on $1/\alpha$ is *exponentially* better. That is, the lemma shows that for a set $X \subseteq \mathbb{F}_2^n$ of size $\alpha \cdot |\mathbb{F}_2|^n$, where $\alpha \in (0, 1]$, there exists a subspace $V \subseteq \mathbb{F}_2^n$ of dimension $\dim(V) \geq n - O(\log^4(1/\alpha))$ such that $V \subseteq 4X$. However, as in the case of Bogolyubov's lemma, we have the problem that the statement is only *existential*.

We thus prove a probabilistic version of the quasi-polynomial Bogolyubov-Ruzsa lemma (see Lemma 3.3) over any field $\mathbb{F} = \mathbb{F}_p$, which asserts that for an α-dense set $X \subseteq \mathbb{F}^n$, there exists a subspace $V \subseteq \mathbb{F}^n$ of dimension $\dim(V) \geq n - O(\log^4(1/\alpha))$ such that for all $v \in V$ it holds that

$$\Pr_{x_1,x_2,x_3 \in \mathbb{F}^n}[x_1, x_2 \in A, x_3, x_4 \in -A] \geq \Omega(\alpha^5) \ ,$$

where $x_4 = v - x_1 - x_2 - x_3$. Furthermore, by combining the techniques above, we show that given a query access to the set X, there is an algorithm that runs in time $\exp(\log^4(1/\alpha)) \cdot \mathrm{poly}\log(1/\delta) \cdot \mathrm{poly}(n)$ and with probability $1 - \delta$ computes a set of vectors $R \subseteq \mathbb{F}^n$ such that $V = \{v \in \mathbb{F}^n : \langle v, r \rangle = 0 \ \forall r \in R\}$.

We are grateful to Tom Sanders for providing us with the argument for showing this lemma, and we provide the proof in Appendix A.

Our local correction lemma. We are now ready to provide an informal statement of our local correction lemma, which builds on the machinery above, and in particular, on the probabilistic quasi-polynomial Bogolyubov-Ruzsa lemma.

Loosely speaking, our local correction allows us to decompose any vector $y \in \mathbb{F}^n$ as a linear combination of the form

$$y = x_1 + x_2 - (x_3 + x_4) + s \ ,$$

where $x_1, x_2, x_3, x_4 \in X$ and $s \in \mathbb{F}^n$ is a *sparse* vector.

Lemma 2.1 (informally stated, see Lemma 3.4)**.** *For a field $\mathbb{F} = \mathbb{F}_p$ and α-dense set $X \subseteq \mathbb{F}^n$, there exists $t \leq 1/\alpha^2$ vectors $b_1, \ldots, b_t \in \mathbb{F}_2^n$ and indices $k_1, \ldots, k_t \in [n]$ satisfying the following. Given a vector $y \in \mathbb{F}_2^n$, let $s = \sum_{j=1}^{t} \langle y, b_j \rangle \cdot \vec{e}_{k_j}$ we have*

$$\Pr_{x_1,x_2,x_3 \in \mathbb{F}^n}[x_1, x_2 \in X, x_3, x_4 \in -X] \geq \Omega(\alpha^5) \ ,$$

where $x_4 = y - s - x_1 - x_2 - x_3$.

Furthermore, given an oracle that computes $1_X(x)$ with probability at least $2/3$, there exists an algorithm that makes $\exp(\log^4(1/\alpha)) \cdot \mathrm{poly}\log(1/\delta) \cdot \mathrm{poly}(n)$ oracle calls and field operations, and with probability at least $1 - \delta$ outputs $b_1, \ldots, b_t$ and $k_1, \ldots, k_t$.

The aforementioned local correction lemmas lie at the heart of our average-case to worst-case reductions, which we discuss next.

2.3 Illustrating example: matrix multiplication

We present a high-level overview of our reductions for matrix multiplication, which illustrates the key ideas that go into the proof. Let ALG be an average-case algorithm that can compute matrix multiplication for an α-fraction of all pairs of matrices $A, B \in \mathbb{F}^n$ in time $T(n)$. We use the *average-case* algorithm ALG to construct a *worst-case* randomized algorithm ALG' that runs in time $O(T(n))$ and computes $A \cdot B$ with high probability for *every* pair of matrices A and B. For simplicity of the exposition, in this overview we make the following assumptions: (1) the algorithm ALG is *deterministic*, (2) the input is a pair (A, B) such that A is a matrix satisfying $\Pr_{B'}[\mathsf{ALG}(A, B') = A \cdot B'] \geq \alpha$, (3) the success rate α is a constant, and (4) the field $\mathbb{F}$ is $\mathbb{F}_2$.

We start by noting two simple facts. First, given the algorithm's (potentially wrong) output $\mathsf{ALG}(A, B)$, we can efficiently check whether the computation is correct using Freivalds' algorithm

(Lemma 4.1). Second, denoting by $X = \{B' \in \mathbb{F}_2^{n\times n} : \mathsf{ALG}(A, B') = A \cdot B'\}$ the set of "good" matrices, we have that if $B \in X$, then the average-case algorithm correctly outputs $\mathsf{ALG}(A, B) = A \cdot B$. Hence, the main challenge is in dealing with the case that $B \notin X$, in which we need to locally correct the value of the multiplication.

Local correction via Bogolyubov's lemma. The first idea is to reduce the problem to the case where the set of good matrices contains a large subspace, and hence admits local correction, as discussed in Section 2.2. Specifically, by the probabilistic Bogolyubov lemma, given X we can choose a subspace $V \subseteq \mathbb{F}_2^{n\times n}$ of matrices, where $\dim(V) \geq n^2 - 1/\alpha^2$, such that for any $B' \in V$, if we sample M_1, M_2, M_3 uniformly at random, then

$$\Pr[M_1, M_2, M_3, M_4 \in X] \geq \alpha^5 \text{ , where } M_4 = B' - M_1 - M_2 - M_3 \text{ .}$$

Note that if the matrices M_1, M_2, M_3, M_4 produced by our sampling are all in the set of good matrices X, then we can self-correct the value of $\mathsf{ALG}(A, B')$ by evaluating $\{\mathsf{ALG}(A, M_i)\}_{i\in[4]}$ and computing the linear combination

$$\sum_{i=1}^{4} \mathsf{ALG}(A, M_i) = \sum_{i=1}^{4} A \cdot M_i = A \cdot (\sum_{i=1}^{4} M_i) = A \cdot B' \text{ .}$$

Note that this event is only guaranteed to occur with probability α^5, which is far smaller than $1/2$. Nevertheless, since we can verify the computation using Freivalds' algorithm, we can boost this probability to be arbitrarily close to 1 by repeating the random sampling step $O(1/\alpha^5)$ times, each time computing $\sum_{i=1}^{4} \mathsf{ALG}(A, M_i)$ and verifying if the obtained result is indeed correct using Freivalds' algorithm. Therefore, if B belongs to the (unknown) subspace V, then the algorithm described above indeed computes $A \cdot B$ with high probability in time $O(T(n)/\mathrm{poly}(\alpha)) = O(T(n))$.

However, the approach above does not work for matrices B that do not lie in the subspace V described above. To deal with this case, our next goal is to "shift" the matrix into the subspace V using low-rank random shifts, which can then be computed efficiently and used for local correction. We describe this procedure next.

Low-rank random matrix shifts. We start by making the following key observation: if we have an arbitrary matrix A, and a matrix $B \in \mathbb{F}^{n\times n}$ of rank k, then their product AB can be computed in time $O(kn^2)$, given a rank-k decomposition of B. Details follow.

To see this, suppose that the first k columns of B denoted by $(B_i)_{i=1}^{k}$, are linearly independent, and for each of the remaining $n - k$ columns $(B_j)_{j=k+1}^{n}$, we know the linear combination $B_j = \sum_{i=1}^{k} d_{i,j} \cdot B_i$ for some coefficients $d_{i,j} \in \mathbb{F}$. We can first multiply A by each of the k linearly independent columns of B. Then, to compute the remaining columns, for each $i = 1, \ldots, k$ let $C_i = A \cdot B_i$ be the i'th column of the matrix $C = AB$, and observe that if $B_j = \sum_{i=1}^{k} d_{i,j} B_i$, then $C_j = A \cdot B_j = A \cdot (\sum_{i=1}^{k} d_{i,j} B_i) = \sum_{i=1}^{k} d_{i,j} \cdot C_i$, which can be computed in $O(kn)$ time for each j. Therefore the total running time of multiplying A by B is $O(kn^2)$.

We are now ready to describe our method for shifting the matrices into the subspace V using low-rank matrices, capitalizing on the observation above. Given the matrix B (that is, possibly, not in V), we sample a random matrix $R_B \in \mathbb{F}^{n\times n}$ of rank $2k$ by randomly choosing $2k$ columns and filling them with uniformly random field elements. Note that with high probability these $2k$ columns

are linearly independent. Then, we let the rest of the columns be random linear combinations of the first $2k$ columns we chose. We observe that if $\dim(V) = n - k$, then

$$\Pr[B + R_B \in V] \geq \frac{1}{2|\mathbb{F}|^k} .$$

If indeed $B + R_B \in V$, then we can compute $A \cdot (B + R_B)$ using the procedure discussed above, by writing $B + R_B$ as a sum of 4 random matrices $B + R_B = M_1 + M_2 + M_3 + M_4$, applying $\mathsf{ALG}(A, M_i)$ for each $i = 1...4$, and using Freivalds' algorithm to efficiently check if the produced output is correct or not.

Note that since we have a lower bound on the probability that $B + R_B$ belongs to the desired subspace, we have an upper bound on the expected number of attempts required until this event occurs. When we obtain such low-rank matrix shifts, which we verify using Freivalds' algorithm, we proceed by computing $A \cdot R_B$. Since R_B is a matrix of rank at most $2k$, the total running time of this will be $O(kn^2)$. Finally, we return

$$\mathsf{ALG}(A, B + R_B) - A \cdot R_B ,$$

which indeed produces the correct answer assuming that $\mathsf{ALG}(A, B + R_B)$ is correct.

Remark 2.2. *The discussion above made the simplifying assumption that the inputs we are getting are pairs (A, B) such that A is a matrix satisfying $\Pr_{B'}[\mathsf{ALG}(A, B') = A \cdot B'] \geq \alpha$. The actual proof require also handling the inputs for which the matrix A does not satisfy this requirement, which is done using similar ideas by applying the local correction procedure first to A and then to B.*

2.4 Beyond matrix multiplication

We conclude the technical overview by briefly sketching some of the key ideas in the rest of our worst-case to average-case reductions, building on the local correction lemmas outlined in Section 2.2. Below we assume that all data structures are deterministic, but by standard techniques this assumption is without loss of generality. We start with the simplest setting, and then proceed to the more involved ones.

Worst-case to average-case reductions for all linear data structure problems. The setting here is the closest to that of matrix multiplication. Let DS_A be an average-case data structure for a linear problem defined by A, where we preprocess an input vector x and the answer to query i is $\langle A_i, x \rangle$, and A_i is the i'th row of A.

Given a vector $y \in \mathbb{F}^n$, we use our local correction lemma to obtain a decomposition of the form $y = x_1 + x_2 - (x_3 + x_4) + v$, where $x_1, x_2, x_3, x_4 \in X$ (i.e., on which $DS_{x_j}(i) = \langle A_i, x_j \rangle$ for all i) and a sparse shift vector $v = \sum_{j=1}^{t} \langle y, b_j \rangle \cdot \vec{e}_{k_j}$. We then preprocess each of the x_j's by applying DS_A to it, and we also compute $\langle A_i, v \rangle$ efficiently by using its sparse representation. The idea is that by the linearity of the problem, we can locally correct according to $\sum_{j=1}^{4} DS_{x_j}(i) + \langle A_i, v \rangle$.

It important to note that, unlike in the setting of matrix multiplication, we cannot use the random low-rank matrix shifts, nor Freivald's algorithm for verification. However, this is where we rely on the sparse subspace decomposition to shift the input into the subspace V implied by the quasi-polynomial Bogolyubov-Ruzsa lemma. In addition, instead of relying on Freivalds' algorithm for verification, here we use the guarantee about the correctness of computation in the subspace V together with the sparsity of the shift vector, which allows us to correct its corresponding contribution via explicit computation. See details in Section 6.1.

Online matrix-vector multiplication (OMV). The online setting of the OMV problem poses several additional challenges. Recall that in the average-case reductions above, the input is a vector $x \in \mathbb{F}_2^n$, each query is a coordinate $i \in [n]$, and the matrix $M \in \mathbb{F}_2^{n \times n}$ is a hard-coded parameter. In the OMV problem, the matrix M is the *input*, the vector x is the *query*, and answer to a query is not a scalar but rather a *vector*. Hence we need to use a two-step local correction where we first decompose the matrix and then decompose the vector. Observe that we can use our additive combinatorics mechanism to preprocess the matrix M and get a description of the subspace V that it asserts, as well as the formula that is required to compute the shift vector s given x, but the problem is that here we cannot preprocess x, as it arrives online. Thus, in the query phase, when the algorithm receives x, we want to find the decomposition $x = x_1 + x_2 + x_3 + x_4 + s$. We then compute the shift vector s, and then sample x_i's whose sum is $x - s$. However this leaves us with the task of checking that all of the x_i's are computed correctly. To this end, we rely on a generalization of *small-bias sample spaces* to finite fields in order to obtain an efficient verification procedure. See details in Section 5.

Weak-average-case reductions. As discussed in the introduction, in the setting of weak-average-case we cannot expect a reduction for all linear problems. In turn, this leads to substantially different techniques. We concentrate on the multivariate polynomial evaluation problem. Here, we are given a polynomial $p\colon \mathbb{F}^m \to \mathbb{F}$ of degree d, where for simplicity, in this overview we fix the parameters $d = \log(n)$, $|\mathbb{F}| = \mathrm{poly}(\log(n))$, and $m = \log(n)/\log\log(n)$, so that we encode n field elements using a codeword of length $\mathrm{poly}(n)$, and the distance is $1 - dm/|\mathbb{F}| > 0.99$. The polynomial is given as input by its $n = d^m$ coefficients, the queries are of the form $x \in \mathbb{F}^m$, and the goal is to output $p(x)$. The key difficulty here, is that for small values of the average-case rate $\alpha > 0$, we need to be able to deal with the *list decoding regime* (see discussion in Section 1.1.4).

The first step is to rely on our additive combinatorics local correction tools similarly as in the OMV reduction. Here the idea is to preprocess the polynomial p and obtain a decomposition of the form $p = p_1 + p_2 + p_3 + p_4 + s$, where again s is a sparse shift-vector. We then construct a data structure for each p_i. However, since we cannot process the queries $x \in \mathbb{F}^m$, we are left with the task of locally correcting the noisy polynomials $\{p_i\}$. If a polynomial p_i is only slightly corrupted (i.e., within the unique decoding regime), we can easily locally correct it without using any preprocessing. However, we also need to deal with noisy polynomials p_i in the *list decoding regime* in which only α-fraction of the points are evaluated correctly, for an arbitrarily small α.

We overcome the difficulty above by capitalizing the preprocessing power of the data structure. Namely, we will show how to boost the success probability from the list-decoding regime to the unique-decoding regime, in which case we can perfectly correct the polynomial via the local correction of the Reed–Muller code. The key idea is that by the generalized Johnson bound, there is only a list of $O(1)$ codewords that agree with the average-case data structure on at least $\alpha/2$-fraction of the points. We thus fix a reference point $w \in \mathbb{F}^m$ and explicitly compute the correct value of $p_i(w)$. Next, we sample a random point r and query the points of line $\ell_{x,w}$ incident to r and the reference point z. Then, we consider the list (of size $O(1)$) of all low-degree univariate polynomials that agree with the queried points on $\ell_{x,w}$, and trim the list by removing each polynomial that does not agree on the reference point. Using the sampling properties of lines in multivariate polynomials, we can show that answering accordingly to the remaining polynomials in the list would yield the right value with high probability.

3 Additive combinatorics toolbox

In this section, we provide a toolbox for locally correcting vectors using techniques from additive combinatorics. The toolkit will play a key technical role in all of our worst-case to average-case reductions.

Throughout this section we fix a finite field $\mathbb{F}$. For simplicity, we set $\mathbb{F} = \mathbb{F}_p$ for a prime number p. However, we remark that the following results hold for any finite field, with only a negligible change in parameters (see discussions in relevant places below). Recall that the sumset of a set X is defined as $X + X = \{x_1 + x_2 : x_1, x_2 \in X\}$, and, similarly, $t \cdot X = \{x_1 + \ldots + x_t : x_1, \ldots, x_t \in X\}$ for an integer $t \geq 1$.

Let $X \subset \mathbb{F}^n$ be a subset of size $|X| = \alpha \cdot |\mathbb{F}|^n$. As we outlined in Section 2, our goal is to decompose any vector $y \in \mathbb{F}^n$ as a linear combination of the form

$$y = x_1 + x_2 - (x_3 + x_4) + s \ ,$$

where $x_1, x_2, x_3, x_4 \in X$, and $s \in \mathbb{F}^n$ is a sparse vector.

Towards this end, we will need additive combinatorics lemmas that will allow us to find a large subspace $V \subseteq 2X - 2X$, so that any vector $v \in V$ can be written as $v = x_1 + x_2 - (x_3 + x_4)$. Crucially, we will show that we can efficiently sample such a decomposition and verify membership in the subspace V.

3.1 Probabilistic and quasi-polynomial Bogolyubov-Ruzsa lemmas

A natural starting point for obtaining a subspace as discussed above is via *Bogolyubov's lemma*, which states that for any subset $X \subseteq \mathbb{F}_2^n$ of density $|X|/2^n \geq \alpha$, there exists a subspace $V \subseteq 4X$ of dimension at least $n - \alpha^{-2}$. However, in addition to minor issues such as being restricted to the field $\mathbb{F}_2$, there are some fundamental problems with using Bogolyubov's lemma for local correction. Most importantly for our application is that while Bogolyubov's lemma asserts that *there exists* a decomposition of each input into a sum of four elements in X, it does not tell us how to obtain this decomposition.

Hence, we further show that each vector $v \in V$ has many "representations" of a sum of 4 elements from X. This way, for any $v \in V$ we can efficiently sample a representation $v = x_1 + x_2 + x_3 + x_4$, where each $x_i \in X$. We refer to this statement as the probabilistic Bogolyubov lemma. To make the following discussion precise, we shall need the following notation.

Given a set $X \subseteq \mathbb{F}^n$, we denote by $1_X \colon \mathbb{F}^n \to \{0,1\}$ the indicator function of the set X. The convolution of two boolean functions f and g we denote by $(f * g)(x) = \mathbb{E}_y[f(y)g(x-y)]$. The Fourier expansion of a function $f \colon \mathbb{F}^n \to \mathsf{C}$ is given by $f(x) = \sum_{r \in \mathbb{F}^n} \hat{f}(r) \cdot \chi_r(x)$, where the Fourier coefficients of f are defined as $\hat{f}(r) = \langle f, \chi_r \rangle = \mathbb{E}_x[f(x) \cdot \overline{\chi_r(x)}]$, with $\chi_r(v) = \omega^{\langle v, r \rangle}$ and $\omega = e^{\frac{2\pi i}{p}}$ is the p'th root of unity. In particular for convolution of two functions we have $(f * g)(x) = \sum_r \hat{f}(r)\hat{g}(r)\chi_r(x)$.

Lemma 3.1 (Probabilistic Bogolyubov lemma)**.** *Let $\mathbb{F} = \mathbb{F}_p$ be a prime field, and let $X \subseteq \mathbb{F}^n$ be a set of size $|X| = \alpha \cdot |\mathbb{F}|^n$ for some $\alpha \in (0,1]$. Let $R = \{r \in \mathbb{F}^n \setminus \{0\} : \left|\hat{1}_X(r)\right| \geq \alpha^{3/2}\}$, and let $V = \{v \in \mathbb{F}^n : \langle v, r \rangle = 0 \ \forall r \in R\}$ be a linear subspace defined by R. Then $|R| \leq 1/\alpha^2$, and for all $v \in V$ it holds that*

$$\Pr_{x_1,x_2,x_3}[x_1, x_2, x_3, v - x_1 - x_2 - x_3 \in X] \geq \alpha^5 \ .$$

Proof. Note first that by Parseval's identity we have $\alpha = \langle 1_X, 1_X \rangle = \|1_X\|_2^2 = \sum_r \left|\hat{1}_X(r)\right|^2$. In particular, for $R = \{r \in \mathbb{F}^n \setminus \{0\} : \left|\hat{1}_X(r)\right|^2 > \alpha^3\}$ we have $|R| \leq \frac{\alpha}{\alpha^3} = \frac{1}{\alpha^2}$. Furthermore, we have

$$\sum_{r \in \mathbb{F}^n \setminus (R \cup \{0\})} \left|\hat{1}_X(r)\right|^4 \leq \alpha^3 \cdot \sum_{r \in \mathbb{F}^n \setminus (R \cup \{0\})} \left|\hat{1}_X(r)\right|^2 \leq \alpha^3(\alpha - \alpha^2) \leq \alpha^4 - \alpha^5 \ ,$$

where the second inequality uses that $\sum_r \left|\hat{1}_X(r)\right|^2 = \alpha$, and $\left|\hat{1}_X(0)\right|^2 = \alpha^2$.

Noting that for every $v \in V$ we have $\chi_r(v) = \omega^{\langle v,r \rangle} = \omega^0 = 1$ for all $r \in R$, it follows that

$$\begin{aligned}
\Pr_{x_1,x_2,x_3 \in \mathbb{F}^n}[x_1, x_2, x_3, v - x_1 - x_2 - x_3 \in V] &= (1_X * 1_X * 1_X * 1_X)(v) \\
&= \sum_{r \in \mathbb{F}^n} (\hat{1}_X(r))^4 \chi_r(v) \\
&= \left|\hat{1}_X(0)\right|^4 \chi_0(v) + \sum_{r \in R} \left|\hat{1}_X(r)\right|^4 \chi_r(v) \\
&\quad + \sum_{r \in \mathbb{F}^n \setminus (R \cup \{0\})} \left|\hat{1}_X(0)\right|^4 \chi_r(v) \\
&\geq \alpha^4 + |R| \cdot \alpha^6 - (\alpha^4 - \alpha^5) \\
&\geq \alpha^5 \ ,
\end{aligned}$$

as required. □

In fact, the foregoing lemma suffices for our application for worst-case to average-case reductions where the success rate α is *a constant*. However, to also allow for success rates that tend to zero, we shall need a much stronger statement of the form of the quasi-polynomial Bogolyubov-Ruzsa lemma, due to Sanders [San12] (see also [Lov15, BRTW14]), which admits an exponentially better dependency on α, albeit without the efficient sampling property.

Lemma 3.2 (Quasi-polynomial Bogolyubov-Ruzsa lemma [San12])**.** *Let $\mathbb{F} = \mathbb{F}_p$ be a prime field, and let $X \subseteq \mathbb{F}^n$ be a set of size $\alpha \cdot |\mathbb{F}|^n$ for some $\alpha \in (0, 1]$. There exists a subspace $V \subseteq \mathbb{F}^n$ of dimension $\dim(V) \geq n - O(\log^4(1/\alpha))$ such that $V \subseteq 2X - 2X$.*

The caveat, however, is that while in Lemma 3.2 the codimension of V is only *polylogarithmic* in $1/\alpha$ (as opposed to polynomial, as in the probabilistic Bogolyubov lemma), it only guarantees that for each $v \in V$ *there exist* $x_1, x_2, x_3, x_4 \in X$ such that $x_1 + x_2 + x_3 + x_4 = v$.

Hence, we further show that each vector $v \in V$ has many "representations" in $2X - 2X$. In particular, for any $v \in V$ we can efficiently sample a representation $v = x_1 + x_2 - x_3 - x_4$, where each $x_i \in X$. We are grateful to Tom Sanders for providing us with a modification of his proof that admits a probabilistic version of the quasi-polynomial Bogolyubov-Ruzsa lemma. Furthermore, we rely on the Goldreich-Levin algorithm and the techniques in [BRTW14] to obtain an efficient algorithm for verifying membership in the implied subspace. This yields the main technical tool that underlies our local correction paradigm.

Lemma 3.3 (Probabilistic quasi-polynomial Bogolyubov-Ruzsa lemma)**.** *Let $\mathbb{F} = \mathbb{F}_p$ be a prime field, and let $A \subseteq \mathbb{F}^n$ be a set of size $|A| = \alpha \cdot |\mathbb{F}|^n$, for some $\alpha \in (0, 1]$. Then, there exists a subspace $V \subseteq \mathbb{F}^n$ of dimension $\dim(V) \geq n - O(\log^4(1/\alpha))$ such that for all $v \in V$ it holds that*

$$\Pr_{a_1,a_2,a_3 \in \mathbb{F}^n}[a_1, a_2 \in A, a_3, a_4 \in -A] \geq \Omega(\alpha^5) \ ,$$

where $a_4 = v - a_1 - a_2 - a_3$. Furthermore, given a query access to the set A, there is an algorithm that runs in time $\exp(\log^4(1/\alpha)) \cdot \operatorname{poly}\log(1/\delta) \cdot \operatorname{poly}(n)$ and with probability $1-\delta$ computes a set of vectors $R \subseteq \mathbb{F}^n$ such that $V = \{v \in \mathbb{F}^n : \langle v, r\rangle = 0 \; \forall r \in R\}$.

We defer the proof of Lemma 3.3 to Appendix A.

3.2 Local correction lemma

Using the probabilistic quasi-polynomial Bogolyubov-Ruzsa lemma (i.e., Lemma 3.3), for any vector $v \in V$ we can efficiently sample $x_1, x_2, x_3, x_4 \in X$ such that we can write

$$v = x_1 + x_2 - (x_3 + x_4) \; .$$

However, we need to be able to handle any vector $y \in \mathbb{F}^n$, and not just vectors in the subspace V. Towards that end, we show that since the subspace implied by the probabilistic quasi-polynomial Bogolyubov-Ruzsa lemma is of large dimension (i.e., of dimension $\dim(V) \geq n - O(\log^4(1/\alpha))$), we can decompose any vector $y \in \mathbb{F}^n$ as a linear combination of the form

$$y = x_1 + x_2 - (x_3 + x_4) + s \; ,$$

where $x_1, x_2, x_3, x_4 \in X$ and $s \in \mathbb{F}^n$ is a *sparse* vector. We stress that the sparsity of the decomposition is essential to our applications, as we cannot locally correct the shift part of the decomposition, and instead we need to compute it explicitly.

The above captures our local correction lemma which will be used throughout this paper.

Lemma 3.4 (Efficient local correction)**.** *Let $\mathbb{F} = \mathbb{F}_p$ be a prime field, and let $X \subseteq \mathbb{F}^n$ be a set of size $|X| = \alpha \cdot |\mathbb{F}|^n$, for some $\alpha \in (0, 1]$. Then, there exists a non-negative integer $t \leq O(\log^4(1/\alpha))$, a collection of t vectors $B = \{b_1, \ldots, b_t \in \mathbb{F}^n\}$, and t indices $k_1, \ldots, k_t \in [n]$ satisfying the following:*

Given a vector $y \in \mathbb{F}^n$, define $s = \sum_{j=1}^{t} \langle y, b_j\rangle \cdot \vec{e}_{k_j}$ where $(\vec{e}_i)_{i\in[n]}$ is the standard basis. Then

$$\Pr_{x_1,x_2,x_3\in\mathbb{F}^n}[x_1, x_2 \in X, x_3, x_4 \in -X] \geq \Omega(\alpha^5) \; ,$$

where $x_4 = y - s - x_1 - x_2 - x_3$.

Furthermore, suppose we have a randomized membership oracle O_X that for every input $x \in \mathbb{F}^n$, computes the indicator $1_X(x)$ correctly with probability at least $2/3$. Then, there exists an algorithm that makes $\exp(\log^4(1/\alpha)) \cdot \operatorname{poly}\log(1/\delta) \cdot \operatorname{poly}(n)$ oracle calls to O_X, performs $\exp(\log^4(1/\alpha)) \cdot \operatorname{poly}\log(1/\delta) \cdot \operatorname{poly}(n)$ field operations, and with probability at least $1-\delta$ returns vectors $b_1, \ldots, b_t \in \mathbb{F}^n$ and indices $k_1, \ldots, k_t \in [n]$ as described above.

Proof. Fix a set $X \subseteq \mathbb{F}^n$ of size $|X| = \alpha \cdot |\mathbb{F}|^n$ for some $\alpha \in (0, 1]$. By applying Lemma 3.3, we obtain a subspace $V \subseteq \mathbb{F}^n$ of dimension $\dim(V) = n - t$ for $t = O(\log^4(1/\alpha))$. Let $R \subseteq \mathbb{F}_2^n \setminus \{\vec{0}\}$ be a set of vectors in $\mathbb{F}^n$ of size t such that $V = \{v \in \mathbb{F}_2^n : \langle v, r\rangle = 0 \; \forall r \in R\}$. Indeed, we can let R be a set of t linearly independent vectors in $V^\perp$.

By writing the vectors of R in a matrix and diagonalizing the matrix, we obtain: (1) a set of vectors $B = \{b_1, \ldots, b_t \in \mathbb{F}_2^n\}$ such that $\mathsf{span}(B) = \mathsf{span}(R)$, and (2) the corresponding pivot indices $k_1, \ldots, k_t \in [n]$ such that $b_j[k_j] = 1$ and $b_j[k_{j'}] = 0$ for all $j \neq j'$.

Given a vector $y \in \mathbb{F}^n$, define $s = \sum_{j=1}^{t} \langle y, b_j \rangle \cdot \vec{e}_{k_j}$, where $(\vec{e}_i)_{i\in[n]}$ is the standard basis, and let $v = y - s$. It is straightforward to verify that $v \in V$. Then for any $j \in [t]$ we have

$$\langle v, b_j \rangle = \langle y, b_j \rangle - \sum_{j=1}^{t} c_j \cdot \left\langle \vec{e}_{k_j}, b_j \right\rangle \overset{(*)}{=} \langle y, b_j \rangle - c_j \cdot \left\langle \vec{e}_{k_j}, b_j \right\rangle \overset{(**)}{=} \langle y, b_j \rangle - \langle y, b_j \rangle = 0 \ ,$$

where (*) is because $\left\langle \vec{e}_{k_{j'}}, b_j \right\rangle = b_j[i_{j'}] = 0$ for $j \neq j'$, and (**) is because $\left\langle \vec{e}_{k_j}, b_j \right\rangle = b_j[i_j] = 1$.

Now, since $v \in V$, by the guarantees of Lemma 3.3 it follows that

$$\Pr_{x_1,x_2,x_3\in\mathbb{F}^n}[x_1 \in X, x_2 \in X, x_3 \in -X, v - x_1 - x_2 - x_3 \in -X] \geq \Omega(\alpha^5) \ .$$

For the furthermore part, note that we can boost the success probability of the membership oracle O_X. That is, given a query x we can decide if $x \in X$ with confidence $1 - \frac{1}{t}$ be repeatedly calling it $O(\log(t))$ times and taking the majority vote. In particular, for $t = \exp(\log^4(1/\alpha)) \cdot \operatorname{poly}\log(1/\delta) \cdot \operatorname{poly}(n)$, by making $O(\operatorname{poly}\log(1/\alpha) + \log(1/\delta) + \log(n))$ calls to $O_X(x)$ for each element $x \in \mathbb{F}^n$ we need to query, we may assume that all queries output whether $x \in X$ or not correctly.

The furthermore part of the lemma follows immediately from the computational guarantees of Lemma 3.3 together with the diagonalization procedure described above. □

4 Worst-case to average-case reductions for matrix multiplication

We prove the worst-case to average-case reduction for matrix multiplication problem in this section. Let's restate Theorem 1 below.

Theorem 1. *Let $\mathbb{F} = \mathbb{F}_p$ be a prime field, $n \in \mathbb{N}$, and $\alpha := \alpha(n) \in (0, 1]$. Suppose that there exists an algorithm ALG that, on input two matrices $A, B \in \mathbb{F}^{n\times n}$ runs in time $T(n)$ and satisfies*

$$\Pr[\mathsf{ALG}(A, B) = A \cdot B] \geq \alpha \, ,$$

where the probability is taken over the random inputs $A, B \in \mathbb{F}^{n\times n}$ and the randomness of ALG.

- *If $|\mathbb{F}| \leq 2/\alpha$, then there exists a randomized algorithm ALG' that for* every *input $A, B \in \mathbb{F}^{n\times n}$ and $\delta > 0$, runs in time $\frac{\exp(O(\log^5(1/\alpha)))}{\delta} \cdot T(n)$ and outputs AB with probability at least $1 - \delta$.*
- *If $|\mathbb{F}| \geq 2/\alpha$, then there exists a randomized algorithm ALG' that for* every *input $A, B \in \mathbb{F}^{n\times n}$ and $\delta > 0$, runs in time $O(\frac{1}{\delta\cdot\alpha^4} \cdot T(n))$ and outputs AB with probability at least $1 - \delta$.*

We divide the proof into two parts, namely for when $|\mathbb{F}| \geq \alpha/2$ and when $|\mathbb{F}| \leq \alpha/2$. The proof for the former case is given in Section 4.2 and the proof of the latter is given in Section 4.1.

We would like to point out that when the field size is large enough, we can use the standard interpolation techniques for low-degree polynomials to prove the reduction. However, the problem becomes more challenging when the field size is small (say, $\mathbb{F} = \mathbb{F}_2$), and showing the reduction in this case requires novel ideas. We will discuss both cases in detail in the following sections.

4.1 Reduction for matrices over small fields

In this section, we show a worst-case to average-case reduction for matrix multiplication problem over small fields, namely, where $|\mathbb{F}| \leq 2/\alpha$, where α is the success rate of the average-case algorithm. Informally, we will show that if there exists an algorithm that is able to compute the multiplication for a small percentage of matrices, then it is possible to boost this algorithm such that it works for all matrices, without sacrificing the running time too much. Before we proceed with the formal result, we need the following lemma known as Freivalds' algorithm.

Lemma 4.1 (Freivalds' Algorithm [Fre77])**.** *Given matrices $A, B, C \in \mathbb{F}^{n\times n}$ there exist a probabilistic algorithm that verifies whether $A \cdot B = C$ with failure probability 2^{-k} where the algorithm runs in $O(kn^2)$.*

Throughout the proof, we will use Freivalds' algorithm to verify the result of matrix multiplication instances,

In particular, it suffices to design an algorithm that given two matrices A, B outputs their product with some non-negligible probability $\varepsilon > 0$. By repeating the algorithm $O(1/\varepsilon)$ times, we can boost the probability of outputting the correct answer to a constant arbitrarily close to 1. This is done by applying Freivalds' algorithm on each of the outputs of the algorithm, rejecting incorrect outputs with high probability, and accepting when the correct answer is found.

We now demonstrate the main result of this section, which corresponds to the first case in Theorem 1.

Theorem 4.2. *Let $\mathbb{F} = \mathbb{F}_p$ be a prime field, $n \in \mathbb{N}$, and $\alpha := \alpha(n) \in (0, 1]$. Suppose that there exists an algorithm ALG that, on input two matrices $A, B \in \mathbb{F}^{n\times n}$ runs in time $T(n)$ and satisfies*

$$\Pr[\mathsf{ALG}(A, B) = A \cdot B] \geq \alpha\,,$$

where the probability is taken over the random inputs $A, B \in \mathbb{F}^{n\times n}$ and the randomness of ALG. If $|\mathbb{F}| \leq 2/\alpha$, then there exists a randomized algorithm ALG' that for every *input $A, B \in \mathbb{F}^{n\times n}$ and $\delta > 0$, runs in time $O(\frac{\exp(O(\log^5(1/\alpha)))}{\delta} \cdot T(n))$ and outputs AB with probability at least $1 - \delta$.*

Below we will prove the theorem assuming the algorithm ALG is deterministic, but a straightforward generalization of the proof works for randomized algorithms as well. To proceed with the proof, we first need the following definitions.

Definition 4.3. *Let X be the set of matrices A such that ALG computes their product with matrices B with probability at least $\alpha/2$. More formally*

$$X = \{A : \Pr_B[\mathsf{ALG}(A, B) = A \cdot B] \geq \alpha/2\}\ .$$

Similarly, for each $A \in \mathbb{F}^{n\times n}$, we define Y_A to be the set of matrices B such that given A and B, ALG correctly computes $A \cdot B$. In other words

$$Y_A = \{B : \mathsf{ALG}(A, B) = A \cdot B\}\ .$$

Claim 4.4. *X and Y_A, where $A \in X$, have density at least $\alpha/2$.*

Proof. Let P_A be the random variable $P_A := \Pr_B[\mathsf{ALG}(A,B) = A \cdot B]$. From the definition, it is clear that $\mathbb{E}_A[P_A] \geq \alpha$. Now, by contradiction, if $\Pr_A[P_A \geq \alpha/2] < \alpha/2$ then we have

$$\mathbb{E}_A[P_A] = \mathbb{E}_A \Pr_B[\mathsf{ALG}(A,B) = A \cdot B] < \alpha/2 \cdot 1 + (1 - \alpha/2) \cdot \alpha/2 < \alpha\,.$$

Hence, $\Pr[P_A \geq \alpha/2] \geq \alpha/2$ and X has density greater than or equal to $\alpha/2$. It follows from the definitions of X and Y_A that for all $A \in X$, Y_A has density at least $\alpha/2$. □

Next, we need the following definition.

Definition 4.5. *For a matrix $A \in \mathbb{F}^{n \times n}$ and $k \in [n]$, let $M_A^k = A + L_A^k$ where we define the matrix L_A^k as follows.*

1. *First, choose a random subset S of size k from $[n]$.*
2. *For each $i \in S$ let the i'th row of L_A^k be uniformly random in $\mathbb{F}^n$.*
3. *For all $j \in [n] \setminus S$, let the j'th row of L_A^k be a random linear combination of the rows in S.*

Remark 4.6. *For matrix L_A^k we have that $\mathsf{rk}(L_A^k) \leq k$, because every row indexed by $j \in [n] \setminus S$ is a linear combination of the rows in S.*

Remark 4.7. *If the random rows in S are not linearly independent, we can throw them away and repeat Step 2. It is not hard to see that this event happens only with constant probability, and we can check this in $O(nk^2)$ time.*

The following lemma shows that matrix $M_A^{2k} = A + (L_A^{2k})$ belongs to any subspace of matrices of constant co-dimension k with constant probability.

Lemma 4.8. *Given a matrix $A \in \mathbb{F}^{n \times n}$ and $k \in [n]$, for any subspace $V \subseteq \mathbb{F}^{n \times n}$ of $\dim(V) \geq n - k$ we have*

$$\Pr[M_A^{2k} \in V] \geq \frac{1}{2|\mathbb{F}|^k}\ .$$

Proof. Since V has co-dimension k, it can be defined by k linear constraints on the elements of the matrix as follows.

$$M_A^{2k}(i_0, j_0) + M_A^{2k}(i_1, j_1) + ... + M_A^{2k}(i_r, j_r) = 0\ ,$$

where $r \in [1, n^2]$ denotes the number of coordinates that this constraint depends on. By re-writing M_A^{2k} as a vector $\mathbf{m} \in \mathbb{F}^{n^2}$, we can construct the system of equations $G \cdot \mathbf{m} = \mathbf{0}$ for membership in V. Here, G denotes the matrix of size $k \times n^2$, where each row specifies one single constraint of the aforementioned form. Now, if we diagonalize G using Gaussian elimination, we can re-write the system of equations in the form $G' \cdot \mathbf{m} = \mathbf{0}$ for a matrix G', where for each row a in G', there exists a column b_a which has value 1 in this row and 0 in the other rows.

For all b_a where $a \in [k]$, we consider the coordinate $\mathbf{m}_{b_a}$. The set of these k coordinates $\{\mathbf{m}_{b_1}, \mathbf{m}_{b_2}, ..., \mathbf{m}_{b_k}\}$ corresponds to k pairs of coordinates $\{(c_1, c_1'), (c_2, c_2'), ..., (c_k, c_k')\}$ in the original matrix. Note that these k coordinates belong to at most k rows in M_A^{2k}, and we want to bound the

probability that none of these rows in L_A^{2k} is a linear combination of the other rows. Let Z be the event that all the $2k$ rows are pairwise linearly independent in L_A^{2k}. We have

$$\begin{aligned}\Pr[Z] &= \left(1-\frac{1}{2^{2k}}\right)\left(1-\frac{2}{2^{2k}}\right)\left(1-\frac{4}{2^{2k}}\right)...\left(1-\frac{2^{k-1}}{2^{2k}}\right) \\ &\geq \left(1-\frac{2^{k-1}}{2^{2k}}\right)^k \geq \left(1-\frac{1}{2^{k+1}}\right)^k \\ &\geq 1-\frac{k}{2^{k+1}} \geq \frac{1}{2} \ .\end{aligned}$$

Now, we observe that if Z happens, it means that for all k constraints, there exist a coordinate which we denote by (c_i, c_i') in M_A^{2k} such that none of the other constraints depend on the value of $M_A^{2k}(c_i, c_i')$, and the value of $M_A^{2k}(c_i, c_i')$ is chosen uniformly at random. Hence, this random value is equal to the unique solution which satisfies ith constraint (assuming values of all other coordinates involved in this constraint are determined beforehand) with probability $1/|\mathbb{F}|$. Therefore, the probability that $M_A^{2k} \in V$ is bounded by

$$\begin{aligned}\Pr[M_A^{2k} \in V] &= \Pr[\text{All } k \text{ constraints are satisfied}] \\ &\geq \Pr[Z] \cdot \frac{1}{|\mathbb{F}|^k} \\ &= \frac{1}{2|\mathbb{F}|^k} \ .\end{aligned}$$

□

Proof of Theorem 4.2. To prove this theorem, we design the following algorithm and prove that this algorithm outputs the correct answer for the matrix multiplication problem with high probability.

Algorithm 1 : Matrix multiplication reduction over small fields

Input: ALG, $A, B \in \mathbb{F}^{n\times n}$
Output: $A \cdot B$ Set k to be $O(\log^4(1/\alpha))$.

1. Set k to be $O(\log^4(1/\alpha))$.
2. For matrices A and B, construct the matrices M_A^{2k} and M_B^{2k}.
3. Sample 3 random matrices $R_1, R_2, R_3 \in \mathbb{F}^{n\times n}$ and set $R_4 = R_1 + R_2 - R_3 - A - M_A^{2k}$ so that $A + M_A^{2k} = R_1 + R_2 - R_3 - R_4$.
4. Sample 12 random matrices $S_1^{(t)}, S_2^{(t)}, S_3^{(t)} \in \mathbb{F}^{n\times n}$ and set $S_4^{(t)} = S_1^{(t)} + S_2^{(t)} - S_3^{(t)} - B - M_B^{2k}$ for $t \in \{1,2,3,4\}$, so that $B + M_B^{2k} = S_1^{(t)} + S_2^{(t)} - S_3^{(t)} - S_4^{(t)}$.
5. Compute $O_L = \sum_{t=1}^4 \sum_{s=1}^4 \mathsf{sign}_{t,s}\mathsf{ALG}(R_t, S_s^{(t)})$, where $\mathsf{sign}_{t,s} = -1$ if $\{t,s\} \cap \{1,2\} = 1$, and $\mathsf{sign}_{t,s} = 1$ otherwise.
6. Compute $O = O_L - A \cdot L_B^{2k} - L_A^{2k} \cdot B - L_A^{2k} \cdot R_B^{2k}$.
7. If $O = A \cdot B$ (check using Lemma 4.1), then return O.

Correctness: Let's consider X as it is defined in Definition 4.3. By applying Lemma 3.3 on X, we can conclude that there exists subspace V_X with co-dimension at most $O(\log^4(1/\alpha))$ and the guaranteed properties. On the other hand, by Lemma 4.8 we have that

$$\Pr[M_A^{2k} \in V_X] \geq \frac{1}{2|\mathbb{F}|^{O(\log^4(1/\alpha))}} \ .$$

Assuming $M_A^{2k} \in V_X$, by Lemma 3.3, we have that

$$\Pr_{R_1,R_2,R_3}[R_1, R_2, -R_3, -R_4 \in X] \geq \Omega(\alpha^5) \ .$$

Now, for each R_t with $t \in \{1,2,3,4\}$, we can consider Y_{R_t} using Definition 4.3. Similarly, since each Y_{R_t} has density at least $\alpha/2$, we can apply Lemma 3.3 on Y_{R_t} to define the corresponding subspaces $V_{Y_{R_t}}$. Having these four subspaces, we define $V_Y = V_{Y_{R_1}} \cap V_{Y_{R_2}} \cap V_{Y_{-R_3}} \cap V_{Y_{-R_4}}$. It is not hard to see that since each of the four subspaces has co-dimension $O(\log^4(1/\alpha))$, the co-dimension of V_Y is at most $4 \cdot O(\log^4(1/\alpha))$. Thus, by Lemma 4.8

$$\Pr[M_B^{2k} \in V_Y] \geq \frac{1}{2|\mathbb{F}|^{O(\log^4(1/\alpha))}} \ .$$

Given $M_B^{2k} \in V_Y$, for each $t \in \{1,2,3,4\}$ by Lemma 3.3

$$\Pr_{S_1^{(t)},S_2^{(t)},S_3^{(t)}}[S_1^{(t)}, S_2^{(t)}, -S_3^{(t)}, -S_4^{(t)} \in Y_{R_t}] \geq \Omega(\alpha^5) \ .$$

It is important to note that since L_A^{2k} and L_B^{2k} have rank less than or equal to $2k$, and all linear combinations of their rows are known previously, we can compute the multiplications in Step 6 in time $O(n^2 \cdot \log^4(1/\alpha))$.

Since all the events defined above are independent, we conclude that the algorithm succeeds with the following probability

$$\Pr[\text{Algorithm 1 succeeds}] \geq \frac{\Omega(\alpha^{25})}{O(|\mathbb{F}|^{O(\log^4(1/\alpha)))}} \geq \frac{\Omega(\alpha^{25})}{O(\frac{10}{\alpha})^{O(\log^4(1/\alpha))}} \geq \exp(-\log^5(1/\alpha)) \ .$$

Therefore, by repeating the algorithm $\frac{\exp(\log^5(1/\alpha))}{\delta}$ times, and using Freivalds' algorithm for verification, we obtain an algorithm that solves the matrix multiplication on all instances with probability at least $1-\delta$.

Running time: Th running time of the procedure described above is essentially dominated by $\frac{\exp(\log^5(1/\alpha))}{\delta}$ calls to the weak average case algorithm, and hence the total running time is $\frac{\exp(\log^5(1/\alpha))}{\delta} \cdot T(n)$. In particular, even if the algorithm succeeds on a sub-constant fraction of inputs $\alpha = \exp(\log^{0.199}(n))$, the reduction turns it into an algorithm that works for worst case instances in time $T(n) \cdot n^{o(1)}$. □

4.2 Reduction for matrices over large fields

We now prove case 2 of Theorem 1 in this section. For concreteness, we restate this result.

Theorem 4.9. *Let $\mathbb{F} = \mathbb{F}_p$ be a prime field, $n \in \mathbb{N}$, and $\alpha := \alpha(n) \in (0, 1]$. Suppose that there exists an algorithm ALG that, on input two matrices $A, B \in \mathbb{F}^{n\times n}$ runs in time $T(n)$ and satisfies*

$$\Pr[\mathsf{ALG}(A, B) = A \cdot B] \geq \alpha\,,$$

where the probability is taken over the random inputs $A, B \in \mathbb{F}^{n\times n}$ and the randomness of ALG. If $|\mathbb{F}| \geq 2/\alpha$, then there exists a randomized algorithm ALG' that for every *input $A, B \in \mathbb{F}^{n\times n}$ and $\delta > 0$, runs in time $O(\frac{1}{\delta\cdot\alpha^4} \cdot T(n))$ and outputs AB with probability at least $1 - \delta$.*

Proof. To prove Theorem 4.9, we use the following algorithm and we prove that this algorithm outputs the correct answer for the matrix multiplication problem with high probability.

Algorithm 2 : Matrix multiplication reduction over large fields

Input: ALG, $A, B \in \mathbb{F}^{n\times n}$
Output: $A \cdot B$ Set k to be $O(\log^4(1/\alpha))$.

1. Let X and Y be matrices chosen uniformly at random from $\mathbb{F}^{n\times n}$, and let i, j, and k be chosen uniformly at random from $\mathbb{F}$.
2. Compute $\mathsf{ALG}(A + iX, B + iY), \mathsf{ALG}(A + jX, B + jY)$, and $\mathsf{ALG}(A + kX, B + kY)$.
3. If the computations are correct (check using Lemma 4.1), then compute $A \cdot B$ by interpolating $(A + iX, B + iY), (A + jX, B + jY)$, and $(A + kX, B + kY)$.

Correctness: Again, we prove the result assuming the algorithm ALG is deterministic, but a straightforward generalization of the proof works for randomized algorithms as well. We define the set of good pairs of matrices for ALG as follows.

Definition 4.10. *Let $S \subseteq (\mathbb{F}^{n\times n} \times \mathbb{F}^{n\times n})$ be the set of pairs of matrices such that for all $(M, N) \in S$ we have $\mathsf{ALG}(M, N) = M \cdot N$. More formally*

$$S = \{(M, N) : \mathsf{ALG}(M, N) = M \cdot N, M \in \mathbb{F}^{n\times n}, N \in \mathbb{F}^{n\times n}\}\ .$$

Note that by definition, S has density at least α.

Claim 4.11. *Let $\ell_{X,Y} = \{(A + iX, B + iY) : i \in \mathbb{F}\}$ be the line that passes through (A, B) and is defined by matrices X and Y. Then, with probability $\alpha/2$, at least $\alpha/2$ fraction of pairs of matrices on this line belong to S.*

Proof. Let $P_{(X,Y)}$ be the random variable $P_{(X,Y)} = \Pr_i[\mathsf{ALG}(A+iX, B+iY) = (A+iX)\cdot(B+iY)]$. From the definition, $\mathbb{E}[P_{(X,Y)}] \geq \alpha$. Now, by contradiction, if $\Pr[P_{(X,Y)} \geq \alpha/2] < \alpha/2$ then we have

$$\mathbb{E}[P_{(X,Y)}] = \mathbb{E}_{X,Y}\Pr_i[\mathsf{ALG}(A + iX, B + iY) = (A + iX) \cdot (B + iY)] < \alpha/2 \cdot 1 + (1 - \alpha/2) \cdot \alpha/2 < \alpha\ .$$

Hence, $\Pr[P_{(X,Y)} \geq \alpha/2] \geq \alpha/2$. □

Having Definition 4.10, we can make the following claim.

Claim 4.12. *The three pairs of matrices defined in Step 2 of Algorithm 2 belong to S with probability at least $\alpha^4/16$.*

Proof. Note that in Step 1 of Algorithm 2, we are sampling a line $\ell_{X,Y}$, which by Claim 4.11 has density $\alpha/2$ of good pair of matrices with probability $\alpha/2$. Also, in Step 2, we are sampling three uniformly random pairs on this line. Assuming $\ell_{X,Y}$ is a line with density $\alpha/2$ of good pairs of matrices, with probability at least $(\alpha/2)^3$ these three pairs belong to S. Hence, total probability that we sample three pairs such that they all belong to S is at least $(\alpha/2)^4 = \alpha^4/16$. □

Since matrix multiplication is a polynomial of degree 2 in the entries of the matrices, having 3 pairs where ALG outputs correct answers enables us to interpolate the value of $A \cdot B$. Thus, the algorithm succeeds with probability $O(\alpha^4)$. By repeating the algorithm and verifying the answer using Freivalds' algorithm, one can amplify the success probability to any arbitrary constant.

□

5 Worst-case to average-case reductions for online matrix-vector multiplication

In this section we prove Theorem 3, which we restate below.

Theorem 3. *Let $\mathbb{F} = \mathbb{F}_p$ be a prime field, $n \in \mathbb{N}$, and $\alpha := \alpha(n) \in (0, 1]$. Consider the matrix-vector multiplication problem $OMV_{\mathbb{F}}$ for dimension n, and suppose that for some $\alpha > 0$ it holds that*

$$OMV_{\mathbb{F}} \in \mathsf{DS}\left[\begin{array}{rl} \textit{preprocessing time:} & p \\ \textit{memory used:} & s \\ \textit{query time:} & t \\ \textit{success rate:} & \Pr_{M,v}[\mathsf{DS}_M(v) = Mv] \geq \alpha \end{array}\right].$$

Then for every $\delta > 0$,

$$OMV_{\mathbb{F}} \in \mathsf{DS}\left[\begin{array}{rl} \textit{preprocessing time:} & 4p + \exp(\log^4(1/\alpha)) \cdot \operatorname{poly}\log(1/\delta) \cdot \operatorname{poly}(n) \\ \textit{memory used:} & 4s + O(\log^4(1/\alpha)n) + O(n^2) \\ \textit{query time:} & (4t + n) \cdot \operatorname{poly}(1/\alpha) \cdot \operatorname{poly}\log(1/\delta) \\ \textit{success rate:} & \forall M, v : \Pr[\mathsf{DS}_M(v) = Mv] \geq 1 - \delta \end{array}\right].$$

Remark 5.1 (Large fields)**.** *We would like to point out that this problem is more interesting when the field $\mathbb{F}$ is small. Indeed, if the size of $\mathbb{F}$ is relatively large (say, $|\mathbb{F}| > 2/\alpha$), then we can think of the matrix-vector multiplication as a polynomial of degree at most one in the elements of the vector. Therefore, we can use the standard self-correction techniques for evaluating low-degree polynomials to solve this problem. In particular, given a query $v \in \mathbb{F}^n$ we can sample a line $\ell \in \mathbb{F}^n$ that passes through v, and query two random vectors that belong to ℓ, and compute Mv by interpolating the two queried points.*

Before proceeding with the formal proof of Theorem 3, we informally outline the argument.

Proof sketch: The proof of Theorem 3 relies on Lemma 3.4, which shows that any vector in $\mathbb{F}^n$ can be self-corrected via a linear constraint involving four vectors $x_1, x_2, x_3, x_4 \in X$ and a shift by sparse vector u. This result will be used several times in the proof of Theorem 3, which we explain below.

- First, note that if $\Pr_{M,v}[\mathsf{DS}_M(v) = Mv] \geq \alpha$, then there is a collection $Z \subseteq \mathbb{F}^n$ of size $|Z| \geq \alpha/2 \cdot |\mathbb{F}|^n$ of *good* matrices, i.e., those matrices M on which DS succeeds to compute Mv on some non-negligible fraction of vectors v. More formally, $Z = \{M \in \mathbb{F}^{n\times n} : \Pr_v[\mathsf{DS}_M(v) = Mv] \geq \alpha/2\}$.
- First time we apply Lemma 3.4 on the set $Z \subseteq \mathbb{F}^{n\times n}$. (That is, we identify $n \times n$ matrices with vectors of length $N = n^2$.) Roughly speaking, given an arbitrary matrix M we will apply the lemma so that we can write $M = M_1 + M_2 - M_3 - M_4 + U$, where $M_1, M_2, M_3, M_4 \in Z$ and U is a sparse matrix. By the assumption in Theorem 3 each M_i succeeds on a non-negligible fraction of vectors v.
- Second time Lemma 3.4 will be used with the set $X = X_{M_i}$ of vectors $v \in \mathbb{F}^n$ on which DS outputs $M_i v$ correctly, where M_i is each of the matrices above. Using the lemma we will be able to represent every vector $v \in \mathbb{F}^n$ as $v = x_1 + x_2 - x_3 - x_4 + u$, where the x_j's belong to X_{M_i}, i.e., the data structure outputs $M_i x_j$ correctly for all $j = 1, 2, 3, 4$, and $u \in \mathbb{F}^n$ is a sparse vector.
- In particular, for each of the matrices M_i the data structure computes correctly $M_i x_j$ for all $j = 1, 2, 3, 4$, and $M_i u$ can be computed in the query phase by reading only $O(1)$ columns of M_i, as u is a sparse vector.

Before proceeding with the formal proof of Theorem 3, we need the following definition of small-bias sample spaces, and the theorem regarding their existence.

Definition 5.2 (Small-bias sample spaces)**.** *A sample space S over $\mathbb{F}^n$ is called* ε-biased *if for every $r \in \mathbb{F}^n \setminus \{0\}$ and every $b \in \mathbb{F}$ it holds that*

$$\left| \Pr_{s\in S}[\langle s, r\rangle = b] - \frac{1}{|\mathbb{F}|} \right| \leq \varepsilon \ .$$

In other words, a sample space is ε-biased if it ε-fools every nontrivial linear test, i.e., for any $r \in \mathbb{F}^n \setminus \{0\}$ the distribution of $\langle s, r\rangle$ with s sampled from S is close in distribution to $\langle s, r\rangle$ for a *uniformly random* $s \in \mathbb{F}^n$. When the exact value of ε is not important, e.g., by setting $\varepsilon = 0.1$, we usually call such sample spaces *small-bias sets*. These objects have been introduced in the work of Naor and Naor [NN93], followed by a long line of work culminating in the recent almost optimal construction of Ta-Shma [TS17], who showed an efficient construction of such sets of size $O(n/\varepsilon^{2+o(1)})$ over $\mathbb{F}_2$. For our purposes, even a randomized construction will be sufficient (see, e.g., Corollary 3.3 in [AMN98]).

Theorem 5.3. *For every finite field $\mathbb{F}$, constant $\varepsilon \in [0, 1]$ and $n \in \mathbb{N}$, a random set $S \subseteq \mathbb{F}^n$ of size $O(n \log |\mathbb{F}|)$ is an ε-biased with high probability. For the field $\mathbb{F} = \mathbb{F}_2$, there exists an explicit construction of size $|S| = O(n)$.*

For concreteness, we will take $\varepsilon = 0.1$, which suffices for our application.

We are now ready to prove Theorem 3.

Proof of Theorem 3. For each matrix $M \in \mathbb{F}^{n\times n}$, let DS_M be the weak average-case data structure implied by the hypothesis of the theorem, and denote by $X_M = \{x \in \mathbb{F}^n \colon \mathsf{DS}_M(x) = Mx\}$ the set of vectors on which the data structure outputs the correct answer. Let $Z \subseteq \mathbb{F}^{n\times n}$ be the set of matrices on which the data structure outputs the correct answer on at least $\frac{\alpha}{2}$-fraction of the inputs; that is, $Z = \{M \in \mathbb{F}^{n\times n} \colon |X_M| \geq \frac{\alpha}{2}|\mathbb{F}|^n\}$. By Markov's inequality, we have $|Z| \geq \frac{\alpha}{2} \cdot |\mathbb{F}|^{n^2}$. Observe that given access to DS_M, we can easily construct a probabilistic oracle for approximate membership in Z.

Claim 5.4. *There exists a probabilistic membership oracle O_Z that for any query $M \in \mathbb{F}^{n\times n}$ makes $t = O(1/\alpha)$ calls to $\mathsf{DS}_M(x)$ for uniformly random $x \in \mathbb{F}^n$, compares the result to Mx, and accepts if and only if at least $\alpha/3$ fraction of the calls to $\mathsf{DS}_M(x)$ output the correct answer. The oracle has the following guarantees.*

- *If $M \in Z$, then* $\Pr[O_Z(M) = ACCEPT] > 2/3$.
- *If $|X_M| \leq \frac{\alpha}{4}|\mathbb{F}|^n$ then* $\Pr[O_Z(M) = REJECT] > 2/3$.

Using the weak average-case data structure DS_M, we construct a worst-case data structure DS' as follows. First, we describe the preprocessing stage of the data structure $\mathsf{DS}^{(M)}$.

Preprocessing:

Input: A matrix $M \in \mathbb{F}^{n\times n}$

1. Self-correcting M: Using Lemma 3.4 with probability $1-\delta$ we represent the matrix M as $M = M_1 + M_2 - M_3 - M_4 + U$, where each M_i has a large X_{M_i} and U is a t-sparse matrix for $t = O(\log^4(1/\alpha))$. The running time of this step is $\exp(\log^4(1/\alpha)) \cdot \operatorname{poly}\log(1/\delta) \cdot \operatorname{poly}(n)$
2. Self-correcting x: Then, for each M_i, we apply Lemma 3.4 on $X_{M_i} = \{x \in \mathbb{F}^n \colon \mathsf{DS}_{M_i}(x) = M_i x\}$, and compute a collection of $t \leq O(\log^4(1/\alpha))$ vectors $B_i = \{b_1^{(i)}, \ldots, b_t^{(i)} \in \mathbb{F}^n\}$ and t indices $k_1^{(i)}, \ldots, k_h^{(i)} \in [n]$ that allow us to represent each vector $x = x_1 + x_2 - x_3 - x_4 + u_i$, where u_i has at most $O(\log^4(1/\alpha))$ non-zero elements, and $x_j \in X_{M_i}$ for all $i = 1, 2, 3, 4$.
3. Let $S \subseteq \mathbb{F}^n$ be a small-biased set obtained by taking $O(n)$ uniformly random vertices in $\mathbb{F}^n$. Note that for $\mathbb{F} = \mathbb{F}_2$ we can take the explicit set S from Theorem 5.3 with $\varepsilon = 0.1$.
4. For each $e \in S$ compute the multiplication from the left eM_i of e with each of the M_i, and store the pairs (e, eM_i) in the data structure.

Overall, the data structure stores the following information in the preprocessing step:

- The sparse matrix $U \in \mathbb{F}^{n\times n}$ with at most $O(\log^4(1/\alpha))$ non-zero entries, obtained in Step 1;
- For each M_i, the weak average-case data structure DS_{M_i} for M_i, which outputs $\mathsf{DS}_{M_i}(x) = M_i \cdot x$ correctly on at least $\alpha/4$ fraction of x's.
- For each M_i, the corresponding $t = O(\log^4(1/\alpha))$ vectors $B_i = \{b_1^{(i)}, \ldots, b_t^{(i)} \in \mathbb{F}^n\}$, and t indices $k_1^{(i)}, \ldots, k_t^{(i)} \in [n]$, obtained in Step 2;
- The pairs (e, eM_i) for every vector e in the small-biased set $S \subseteq \mathbb{F}^n$ and for every M_i, obtained in Step 3.

Preprocessing time: The preprocessing time is determined by the time needed to find matrices M_i, and the preprocessing time of the weak-average-case data structures. It is easy to verify that the preprocessing time is bounded by $4p + \exp(\log^4(1/\alpha)) \cdot \operatorname{poly}\log(1/\delta) \cdot \operatorname{poly}(n)$, where p is the preprocessing time of the weak-average-case data structure.

Memory used: In the preprocessing step, we store (1) 4 weak-average-case data structures of size s for each of the matrices M_i, (2) for each matrix M_i we store the collection of $t = O(\log^4(1/\alpha))$ vectors B_i and t indices, (3) a representation of the sparse matrix U using $O(\log^4(1/\alpha)\log(n))$ field elements. (4) The pairs (e, eM_i) for every vector e in the small-biased set S, which is of size $O(n)$. Hence, the total space used is $4s + O(\log^4(1/\alpha)n) + O(n^2)$.

Next we describe the query phase of the worst-case data structure. Recall, for each matrix M_i where $i = 1, 2, 3, 4$, we store the vectors $B_i = \{b_1^{(i)}, \ldots, b_h^{(i)} \in \mathbb{F}^n\}$ and indices $k_1^{(i)}, \ldots, k_h^{(i)} \in [n]$ in out data structure, to compute u_i every vector in $\mathbb{F}^n$ can be written as a linear combination of four vectors in X_{M_i}. The query phase works as follows.

Query phase:

Input: A query $x \in \mathbb{F}^n$

1. For $i \in \{1, 2, 3, 4\}$, sample random $x_1^{(i)}, x_2^{(i)}, x_3^{(i)} \in \mathbb{F}^n$ and let $x_4^{(i)}$ be such that $x = u_i + x_1^{(i)} + x_2^{(i)} - x_3^{(i)} - x_4^{(i)}$.

2. For each matrix M_i and for $j \in \{1, 2, 3, 4\}$, apply $\mathsf{DS}_\alpha^{(M_i)}$ to $x_j^{(i)}$.

3. Verify that $\mathsf{DS}^{M_i}(x_j^{(i)}) = M_i x_j^{(i)}$ using the small biased set S. Specifically, sample $O(\log(1/\delta))$ vectors $e \in S$, and check that

$$\left\langle e, \mathsf{DS}_{M_i}(x_j^{(i)}) \right\rangle = \left\langle eM_i, x_j^{(i)} \right\rangle \ .$$

   If the answer of $\mathsf{DS}_{M_i}(x_j^{(i)})$ outputs the correct answer, then the inner products will all be equal. If $\mathsf{DS}_{M_i}(x_j) \neq M_i x_j^{(i)}$, then a random $e \in S$ will catch an inequality with probability at least 0.4.

4. By repeating the sampling above for $O(\log(1/\delta) \cdot 1/\alpha^5)$ times, for each $i \in \{1, 2, 3, 4\}$ we will find such $x_1^{(i)}, x_2^{(i)}, x_3^{(i)}, x_4^{(i)}$ on which $\mathsf{DS}_{M_i}(x_j^{(i)})$ outputs the correct answer with high probability.

5. Compute $M_i u_i$ directly. Since u_i has at most $O(\log^4(1/\alpha))$ non-zero coordinates, it follows that $M_i u_i$ can be computed in time $O(\log^4(1/\alpha)n)$.

6. For $i \in \{1, 2, 3, 4\}$ compute $M_i y$ by taking $M_i x_1^{(i)} + M_i x_2^{(i)} - M_i x_3^{(i)} - M_i x_4^{(i)} + M_i u_i$.

7. Compute Uy directly. Since U has at most $O(\log^4(1/\alpha))$ non-zero elements, this can be done in time $O(\log^4(1/\alpha) \cdot \log(n))$.

8. Return $My = M_1 y + M_2 y - M_3 y - M_4 y + Uy$.

Correctness: To prove the correctness, we bound the failure probability of the algorithm. Note that the failure of the algorithm only depends on the verification procedure in Step 3. In other words, if $\mathsf{DS}'_M(x) \neq Mx$, then at least for one pair of (i,j) we have that $\mathsf{DS}_{M_i} a(x_j^{(i)}) \neq M_i x_j^{(i)}$, and none of the sampled vectors e has caught this inequality. On the other hand, this event happens with probability at most $0.6^{O(\log(1/\delta))}$, bounding the failure probability to be at most δ.

Query time: The query time consist of the time required to compute $M_i x_j^{(i)}$, the time required to compute $M_i u_i$, and the time needed to compute Uy where $i,j \in \{1,2,3,4\}$. The sampling in Step 1 will be done $O(\log(1/\delta) \cdot 1/\alpha^5)$ times, and for each sampled vectors, DS_{M_i} is applied 4 times. Also, the verification in Step 3 consists of computing the inner product for $O(\log(1/\delta))$ many vectors. Thus, the total query time is equal to

$$\begin{aligned} &O(\log(1/\delta) \cdot 1/\alpha^5) \cdot 4 \cdot (t + O(\log(1/\delta)n) + O(n \log^4(1/\alpha))) \\ &\qquad = (4t+n) \cdot \mathrm{poly}(1/\alpha) \cdot \mathrm{poly}\log(1/\delta) \ . \end{aligned}$$

This completes the proof of Theorem 3. □

6 Worst-case to average-case reductions for data structures

In this section, we show worst-case to average-case reductions in the setting of static data structures. We start by showing a reduction for all linear data structure problems in Section 6.1.

Then, we consider a more powerful type of reductions, which can be used to derive worst-case algorithms from data structures that only satisfy a weak average case condition over both inputs and queries (similarly to the setting of online matrix-vector multiplication). On the negative side, we give a counterexample, showing that general weak-average-case reductions cannot hold for all linear problems. On the positive side, we show that the problem of evaluating a multivariate polynomial admits such a weak-average-case reduction. We stress that as opposed to the online matrix-vector multiplication problem discussed above, the problem of multivariate polynomial evaluation is an example of a non-linear problem admitting such a reduction.

6.1 Average-case reductions for all linear problems

Recall that in the setting of data structures, a linear problem over a field $\mathbb{F}$ is defined by a matrix $A \in \mathbb{F}^{m \times n}$. The input to the data structure is a vector $x \in \mathbb{F}^n$, which is preprocessed into s memory cells. Then, given queries of the form $i \in [m]$, the goal of the data structure is to output $\langle A_i, x \rangle$, where A_i is the i'th row of A. We show a worst-case to average-case reduction for data structures for *all* linear problems.

Remark 6.1. *We note that the presented reduction results in* uniform *data structures. That is, we give an efficient and simple procedure that, given an average-case data structure, creates a worst-case data structure in a black-box way that works for all values of n.*

There is a trivial folklore argument that transforms an average-case data structure into a non-uniform *worst-case data structure as follows. Let $X \subseteq \mathbb{F}^n$ of size $|X| \geq \alpha|\mathbb{F}|^n$ be the set where for a given n, the average-case data computes all queries correctly. By the probabilistic method, there exists $(n/\alpha)\log|\mathbb{F}|$ shifts of X that cover all of $\mathbb{F}^n$. For every n, a non-uniform data structure will remember all of those shifts, and for each shift $s \in \mathbb{F}^n$, it will also remember the*

product As. Now, given an input vector x, in the preprocessing stage, the data structure just stores the index of a shift s such that $x + s \in X$, and in the query phase it reads the index of the shift and, thus, learns As. Now, since $x + s \in X$, we can use the average-case data structure to compute $A(x + s)$, and, finally, compute $Ax = A(x + s) - As$. This results in a non-uniform worst-case data structure whose space complexity and query time differ from those of the average-case data structure by an additive term of $\log((n/\alpha) \log |\mathbb{F}|)$.

Theorem 2. *Let $\mathbb{F} = \mathbb{F}_p$ be a prime field, $\alpha := \alpha(n) \in (0, 1]$, $n, m \in \mathbb{N}$, and a matrix $A \in \mathbb{F}^{m \times n}$. Denote by L_A the linear problem of outputting $\langle A_i, x \rangle$ on input $x \in \mathbb{F}^n$ and query $i \in [m]$. Suppose that*

$$L_A \in \mathsf{DS}\left[\begin{array}{rl} \textit{preprocessing time:} & p \\ \textit{memory used:} & s \\ \textit{query time:} & t \\ \textit{success rate:} & \Pr_{x \in \mathbb{F}^n}[\mathsf{DS}_x(i) = \langle A_i, x \rangle \, \forall i \in [m]] \geq \alpha \end{array}\right].$$

Then for every $\delta > 0$,

$$L_A \in \mathsf{DS}\left[\begin{array}{rl} \textit{preprocessing time:} & p + \exp(\log^4(1/\alpha)) \cdot \operatorname{poly}\log(1/\delta) \cdot \operatorname{poly}(n) \\ \textit{memory used:} & 4s + O(\log^4(1/\alpha) \log(n)) \\ \textit{query time:} & 4t + O(\log^4(1/\alpha) \log(n)) \\ \textit{success rate:} & \forall x \in \mathbb{F}^n \;\; \Pr[\mathsf{DS}'_x(i) = \langle A_i, x \rangle \, \forall i \in [m]] \geq 1 - \delta \end{array}\right].$$

Proof. Consider the data structure DS for the matrix A, implied by the assumption of the theorem. There exists a subset $X \subseteq \mathbb{F}^n$ of size $|X| \geq \alpha |\mathbb{F}|^n$ such that for every input $x \in X$ the data structure answers correctly all queries to the data structure, i.e., $\mathsf{DS}_x(i) = \langle A_i, x \rangle$ for all $i \in [m]$ and $x \in X$.

We design a data structure DS' that outputs correct answers in worst case as follows. Let $x \in \mathbb{F}^n$ be the input to DS'. We start by describing preprocessing and query phases of the data structure.

Worst-case data structure for L_A

Preprocessing: Given $x \in \mathbb{F}^n$ we apply the Lemma 3.4 on x with the set X, and obtain a non-negative integer $t \leq O(\log^4(1/\alpha))$, a vector $v \in \mathbb{F}^n$ with at most t non-zero entries, and $x_1, x_2, x_3, x_4 \in X$ such that

$$x = x_1 + x_2 - x_3 - x_4 + v \; .$$

Furthermore, by Lemma 3.4 such decomposition can be found using $\exp(\log^4(1/\alpha)) \cdot \operatorname{poly}\log(1/\delta) \cdot \operatorname{poly}(n)$ field operations with probability at least $1 - \delta$.
Then, we use the preprocessing algorithm of the average-case data structure on each one of the x_j's to obtain the algorithms $DS_{x_1}(\cdot), DS_{x_2}(\cdot), DS_{x_3}(\cdot), DS_{x_4}(\cdot)$. Finally, we store the sparse shift vector v by storing the t coordinates, and their values. Therefore, the amount of memory used is $4s + O(\log^4(1/\alpha) \log(n))$.

Query phase: Given a query $i \in [m]$, we invoke our four instantiations of the average-case data structure stored in the preprocessing stage and compute $\mathsf{DS}_{x_1}(i), \mathsf{DS}_{x_2}(i), \mathsf{DS}_{x_3}(i), \mathsf{DS}_{x_4}(i)$. We then compute $\langle A_i, v \rangle$ and return

$$\mathsf{DS}_{x_1}(i) + \mathsf{DS}_{x_2}(i) - \mathsf{DS}_{x_3}(i) - \mathsf{DS}_{x_4}(i) + \langle A_i, v \rangle \; .$$

Complexity: The time and amount of memory used follow immediately from the description. Namely, note that applying the local correction lemma, which dominates the time complexity, is done in time $\exp(\log^4(1/\alpha)) \cdot \operatorname{poly}\log(1/\delta) \cdot \operatorname{poly}(n)$. Hence the total preprocessing time is $4p + \exp(\log^4(1/\alpha)) \cdot \operatorname{poly}(n)$.

In terms of memory, we store 4 instances of the average-case data structure DS, where each instance requires s memory cells. In addition we store the sparse vector v, by storing its t non-zero indices and their values. Hence the total memory required is $4s + O(\log^4(1/\alpha) \cdot \log(n))$.

Finally, the bound on the query time consists of 4 queries to the the average case data structure DS, as well as the computation of $\langle A_i, v\rangle$. The latter can be done by reading the description of the t-sparse vector v, and computing their inner product with the corresponding t entries in the i'th row of A. Hence the total query time is $4t + O(\log^4(1/\alpha) \cdot \log(n))$.

Correctness: By Lemma 3.4 we have

$$x = x_1 + x_2 - (x_3 + x_4) + v \ ,$$

where $x_1, x_2, x_3, x_4 \in X$. By the definition of X, this implies that the average-case data structure DS computes these points correctly, hence

$$\mathsf{DS}_{x_1}(i) + \mathsf{DS}_{x_2}(i) - \mathsf{DS}_{x_3}(i) - \mathsf{DS}_{x_4}(i) = \langle A_i, x_1\rangle + \langle A_i, x_2\rangle - \langle A_i, x_3\rangle - \langle A_i, x_4\rangle \ ,$$

Furthermore, we directly compute $\langle A_i, v\rangle$, and hence, by the linearly of the inner product operation, it follows that

$$\langle A_i, x\rangle = \sum_{j=1}^{4} \langle A_i, x_j\rangle + \langle A_i, v\rangle \ .$$

This concludes the proof of Theorem 2. □

6.2 Weak-average-case data structures

In Section 6.1, we showed a worst-case to average-case reduction for all linear problems in the setting of data structures. In the following, we show how to obtain worst-case algorithms starting from a very weak, but natural, notion of average-case reductions that we discuss next.

Recall that in the standard definition of average-case data structures, the algorithm preprocesses its input and is then required to correctly answer all queries for an α-fraction of all possible inputs. However, in many cases (such as in the online matrix-vector multiplication problem), we only have an average-case guarantee on both inputs and queries. In this setting, we should first ask what is a natural notion of an average-case condition.

A strong requirement for an average-case algorithm in this case is to correctly answer *all queries* for at least α-fraction of the inputs. However, a more desirable condition is to require the algorithm to correctly answer on an *average input and query*. This is captured by the following definition.

Definition 6.2. *A* weak average-case *data structure for computing a function* $f\colon \mathbb{F}^n \times Q \to \mathbb{F}^k$ *with success rate* $\alpha > 0$ *receives an input* $x \in \mathbb{F}^n$*, which is preprocessed into* s *memory cells. Then, given a query* $q \in Q$*, the data structure* $\mathsf{DS}_x(q)$ *outputs* $y \in \mathbb{F}^k$ *such that*

$$\Pr_{x\in\mathbb{F}^n, q\in Q}[\mathsf{DS}_x(q) = f(x,q)] \geq \alpha \ .$$

The challenge in this setting is that the errors may be distributed between both the inputs and the queries. On one extreme, the error could be concentrated on selected inputs, and then the data structure computes *all queries* correctly for α-fraction of the inputs. On the other extreme, the error could be spread over all inputs, and then the data structure may only answer α-fraction of the queries on any inputs. Of course, the error could be distributed anywhere in between these two extremes.

Weak-average-case reductions for matrix-vector multiplication. As a first example of the weak-average-case paradigm, we note that our reduction for the online matrix-vector multiplication problem in Section 5 can be cast as a worst-case to weak-average-case reduction. Namely, we start with a data structure that receives a matrix $M \in \mathbb{F}^{n \times n}$ as an input, preprocesses it into s memory cells. Then, on query $v \in \mathbb{F}^n$ the data structure algorithm DS_M satisfies

$$\Pr_{M \in \mathbb{F}^{n \times n}, v \in \mathbb{F}^n}[\mathsf{DS}_M(v) = Mv] \geq \alpha \ .$$

Hence we immediately obtain the following statement.

Theorem 3. *Let $\mathbb{F} = \mathbb{F}_p$ be a prime field, $n \in \mathbb{N}$, and $\alpha := \alpha(n) \in (0, 1]$. Consider the matrix-vector multiplication problem $OMV_{\mathbb{F}}$ for dimension n, and suppose that for some $\alpha > 0$ it holds that*

$$OMV_{\mathbb{F}} \in \mathsf{DS}\left[\begin{array}{rl} \textit{preprocessing time:} & p \\ \textit{memory used:} & s \\ \textit{query time:} & t \\ \textit{success rate:} & \Pr_{M,v}[\mathsf{DS}_M(v) = Mv] \geq \alpha \end{array}\right].$$

Then for every $\delta > 0$,

$$OMV_{\mathbb{F}} \in \mathsf{DS}\left[\begin{array}{rl} \textit{preprocessing time:} & 4p + \exp(\log^4(1/\alpha)) \cdot \operatorname{poly}\log(1/\delta) \cdot \operatorname{poly}(n) \\ \textit{memory used:} & 4s + O(\log^4(1/\alpha)n) + O(n^2) \\ \textit{query time:} & (4t + n) \cdot \operatorname{poly}(1/\alpha) \cdot \operatorname{poly}\log(1/\delta) \\ \textit{success rate:} & \forall M, v : \Pr[\mathsf{DS}_M(v) = Mv] \geq 1 - \delta \end{array}\right].$$

An immediate question is whether it is possible to obtain worst-case to weak-average-case reductions not only for the matrix-vector multiplication problem, but rather for *all* linear problems, as we have in the setting of (standard) average-case data structure. Alas, as we show next, such a general result is impossible.

6.3 Impossibility of weak-average-case reductions for all linear problems

We observe that for weak-average-case data structures, there is a simple counterexample which shows that it is *impossible* to obtain worst-case to weak-average-case reductions for *all* linear problems. Nevertheless, we later show that it is possible to obtain such reductions for specific natural problems beyond matrix-vector multiplication, namely for the (non-linear) problem of multivariate polynomial evaluation.

To see the counterexample, first note that a weak-average-case data structure can be equivalently thought of as a data structure where the answer to each input is a vector, rather than a scalar, and the requirement is that the algorithms on average outputs a partially correct vector. That is, a weak-average-case data structure computes a function $f \colon \mathbb{F}^n \to \mathbb{F}^m$ with success rate $\alpha > 0$ if after the preprocessing, on query $x \in \mathbb{F}^n$ it satisfies that $\Pr_{x \in \mathbb{F}^n, i \in [m]}[\mathsf{DS}(x)_i = f(x)_i] \geq \alpha$.

Weak-average-case circuits. For simplicity, we start with a counterexample for weak-average-case circuits, then extend it to the setting of data structures. Note that the number of linear functions $f\colon \mathbb{F}_2^n \to \mathbb{F}_2^m$ is 2^{nm}. The total number of (not necessarily linear) circuits with g gates is $2^{O(g\log(g))}$ (see, e.g., Lemma 1.12 in [Juk12]). Therefore, by a simple counting argument, a random *linear* function requires a circuit with $g \geq \Omega\left(\frac{mn}{\log(mn)}\right)$ gates.

Now fix a linear function f of complexity at least $\Omega\left(\frac{mn}{\log(mn)}\right)$, and an arbitrarily small constant $\varepsilon > 0$. Let us consider the function $h : \mathbb{F}_2^n \to \mathbb{F}_2^{m/\varepsilon}$ that embeds f as follows: the first m outputs of $h(x)$ compute $f(x) \in \mathbb{F}^m$, and the remaining $m/\varepsilon - m$ outputs are always zeros. Note that h is also a linear function that requires a circuit with at least $\Omega\left(\frac{mn}{\log(mn)}\right)$ gates.

On the other hand, note that the trivial circuit outputting m/ε zeros well-approximates the function h, i.e., it satisfies the weak-average-case with success rate $\alpha = (1-\varepsilon)$. Thus, any worst-case to average-case reduction for all functions in this setting would have to blow up the size of the trivial circuit computing 0 to the size of at least $\Omega\left(\frac{mn}{\log(mn)}\right)$, which is almost the biggest circuit with this given number of inputs and outputs. Therefore, such a reduction would be degenerate.

Weak-average-case data structures. Moving on to the setting of data structures, here there is an issue with such an argument. To see that, recall that we want start by picking a *linear* function that is hard even against *non-linear* data structures. While the number of linear functions is still 2^{mn}, a data structure can compute s *arbitrary* (i.e., not necessarily linear) functions in the preprocessing stage. The problem is that even one such function gives a data structure 2^{2^n} possibilities which is already larger than the number of *linear* functions, and so the counting argument here doesn't work.

Nevertheless, we can still get essentially the same result for data structures by the following argument. Let $C(n,m)$ be the complexity of the hardest data structure for a *linear* problem. Then, again, we take the hardest linear function from n bits to m bits, and extend it to a function with m/ε output bits (where $m/\varepsilon - m$ outputs are constant zeros). The worst-case complexity of this function is at least $C(n,m)$, while the average-case complexity is 0. Hence, every worst-case to average-case reduction will blow up the size from 0 to $C(n,m)$. Since every linear function can be computed by a data structure of size $C(n, m/\varepsilon)$ without any reduction, such a reduction is also degenerate.

6.4 Weak-average-case reductions for multivariate polynomial evaluation

Our main result in the weak-average-case setting is a worst-case to weak-average-case reduction for data structures computing the (non-linear) problem of multivariate polynomial evaluation. In this problem, the input is a polynomial $q\colon \mathbb{F}^m \to \mathbb{F}$ of total degree at most d, given as its coefficients. That is, the length of the input is $n = \binom{m+d}{d}$. Given the input polynomial, it is preprocessed, and then in the query phase the goal is to respond to each query $x \in \mathbb{F}^m$ with the value $q(x)$. We restate and prove Theorem 4 below.

Theorem 4. *Let $\mathbb{F} = \mathbb{F}_p$ be a prime field, $\alpha \coloneqq \alpha(n) \in (0,1]$, and let $m, d \in \mathbb{N}$ be parameters. Consider the problem $\mathsf{RM}_{\mathbb{F},\mathsf{m},\mathsf{d}}$ of evaluating polynomials of the form $q\colon \mathbb{F}^m \to \mathbb{F}$ of total degree d (i.e., the problem of evaluating the Reed-Muller encoding of block length $n = \binom{m+d}{d}$).*

Suppose that

$$\mathsf{RM}_{\mathbb{F},\mathsf{m},\mathsf{d}} \in \mathsf{DS}\left[\begin{array}{rl} \textit{preprocessing time:} & p \\ \textit{memory used:} & s \\ \textit{query time:} & t \\ \textit{success rate:} & \Pr_{q,x}[\mathsf{DS}_q(x)] \geq \alpha \end{array}\right].$$

Then

$$\mathsf{RM}_{\mathbb{F},\mathsf{m},\mathsf{d}} \in \mathsf{DS}\left[\begin{array}{rl} \textit{preprocessing time:} & p + \exp(\log^4(1/\alpha)) \cdot \mathrm{poly}(n) \\ \textit{memory used:} & 4s + O(\log^4(1/\alpha)\log(n)) \\ \textit{query time:} & O(|\mathbb{F}|^2 \cdot t + |\mathbb{F}|\log^4(1/\alpha) + |\mathbb{F}|\log(n)) \\ \textit{success rate:} & \forall q, x : \Pr[\mathsf{DS}_q(x) = q(x)] > 1 - O\left(\sqrt{\frac{d}{|\mathbb{F}|}}\right) \end{array}\right].$$

Theorem 4 follows from the following two lemmas.

Lemma 6.3. *Let $\mathbb{F}$ be a prime field, and $d \leq |\mathbb{F}|/10$. Suppose that for $\alpha > 2\sqrt{\frac{d}{|\mathbb{F}|}}$ we have*

$$\mathsf{RM}_{\mathbb{F},\mathsf{m},\mathsf{d}} \in \mathsf{DS}\left[\begin{array}{rl} \textit{preprocessing time:} & p \\ \textit{memory used:} & s \\ \textit{query time:} & t \\ \textit{success rate:} & \Pr_{q,x}[\mathsf{DS}_q(x) = q(x)] \geq \alpha \end{array}\right].$$

Then

$$\mathsf{RM}_{\mathbb{F},\mathsf{m},\mathsf{d}} \in \mathsf{DS}\left[\begin{array}{rl} \textit{preprocessing time:} & 4p + \exp(\log^4(1/\alpha)) \cdot \mathrm{poly}\log(1/\delta) \cdot \mathrm{poly}(n) \\ \textit{memory used:} & 4s + O(\log^4(1/\alpha)) \\ \textit{query time:} & 4|\mathbb{F}| \cdot t + O(\log^4(1/\alpha))) + O(\log(n) \\ \textit{success rate:} & \forall q \textit{ with } deg(q) \leq d\colon \Pr_x[\mathsf{DS}_q(x) = q(x)] \geq 1 - O\left(\sqrt{\frac{d}{\alpha \cdot |\mathbb{F}|}}\right) \end{array}\right].$$

Lemma 6.4. *Let $\mathbb{F}$ be a prime field, and $d \leq |\mathbb{F}|/10$. Suppose that for $\gamma < 0.1$ we have*

$$\mathsf{RM}_{\mathbb{F},\mathsf{m},\mathsf{d}} \in \mathsf{DS}\left[\begin{array}{rl} \textit{preprocessing time:} & p \\ \textit{memory used:} & s \\ \textit{query time:} & t \\ \textit{success rate:} & \forall q \textit{ with } deg(q) \leq d\colon \Pr_x[\mathsf{DS}_q(x) = q(x)] \geq 1 - \gamma \end{array}\right].$$

Then

$$\mathsf{RM}_{\mathbb{F},\mathsf{m},\mathsf{d}} \in \mathsf{DS}\left[\begin{array}{rl} \textit{preprocessing time:} & p \\ \textit{memory used:} & s \\ \textit{query time:} & |\mathbb{F}| \cdot t \\ \textit{success rate:} & \forall q \textit{ with } deg(q) \leq d, \forall x \in \mathbb{F}^n\colon \Pr[\mathsf{DS}_q(x) = q(x)] \geq 1 - 4\gamma \end{array}\right].$$

Before proceeding with the proofs of the lemmas above, we will need the following proposition.

Proposition 6.5. *Let $\mathbb{F}$ be a prime field, $d \leq |\mathbb{F}|/10$, and let $\alpha > 2\sqrt{\frac{d}{|\mathbb{F}|}}$. Let $n = \binom{d+m}{m}$ be the input length—the number of coefficients in a polynomial $q\colon \mathbb{F}^m \to \mathbb{F}$ of total degree at most d,*

Let DS be a data structure for $\mathsf{RM}_{\mathbb{F},\mathsf{m},\mathsf{d}}$ with preprocessing time p, that stores s field elements, and has query time t. Then there exists another data structure DS' for $\mathsf{RM}_{\mathbb{F},\mathsf{m},\mathsf{d}}$ with preprocessing time $p + n$, that stores $s + m + 1$ field elements, has query time $|\mathbb{F}|t$, and satisfies the following guarantee for all input polynomials q of degree at most d.

$$\textit{If } \Pr_x[\mathsf{DS}_q(x) = q(x)] \geq \alpha, \textit{ then } \Pr_x[\mathsf{DS}'_q(x) = q(x)] \geq 1 - \sqrt{\frac{d}{\alpha \cdot |\mathbb{F}|}}.$$

We emphasize that in the claim above the data structures do not depend on the polynomial q. The proposition says that if q is an input such that DS outputs the correct evaluation $q(x)$ for at least α fraction of the queries x, then DS' (which also does not depend on any particular q) succeeds on $1 - \sqrt{\frac{d}{\alpha \cdot |\mathbb{F}|}}$ fraction of the same input q.

Proof of Proposition 6.5. Denote by DS the data structure for $\mathsf{RM}_{\mathbb{F},\mathsf{m},\mathsf{d}}$ that outputs the correct answer for at least α fraction of inputs to q. Below we describe the data structure DS'.

Data Structure DS'

Preprocessing: Given the polynomial q of degree at most d

1. Run the preprocessing procedure for DS on the input q.
2. Choose a random *reference point* $\vec{w} \in \mathbb{F}^m$ and compute $q(\vec{w})$.
3. Store $\vec{w}$ and $q(\vec{w})$ in the memory.

Query: Given a query $\vec{x} \in \mathbb{F}^m$

1. Consider the line $\ell_{\vec{x},\vec{w}} = \{\vec{x} + r(\vec{w} - \vec{x}) : r \in \mathbb{F}\}$ going through $\vec{x}$ and $\vec{w}$.
2. Use the query algorithm of DS to compute $(\mathsf{DS}_q(z) : z \in \ell_{\vec{x},\vec{w}})$.
3. Let $Q = \{q_1, q_2, \ldots, q_k\}$ be all the univariate polynomials of degree at most d that agree with $(\mathsf{DS}_q(z) : z \in \ell_{\vec{x},\vec{w}})$ on at least $\alpha/2$ fraction of points in $\ell_{\vec{x},\vec{w}}$. (It is possible that $Q = \emptyset$.)
4. Use the value $q(\vec{w})$ from the preprocessing phase, and let $Q' = \{q' \in Q : q'(\vec{w}) = q(\vec{w})\}$. (It is possible that $Q' = \emptyset$.)
5. Choose $q' \in Q'$ arbitrarily and output $q'(\vec{x})$.

Next, we claim that if $\alpha > 2\sqrt{\frac{d}{|\mathbb{F}|}}$, then for at least $1 - O\left(\sqrt{\frac{d}{|\mathbb{F}\alpha|}}\right)$ fraction of the queries $\vec{x}$ the query phase correctly outputs $q(\vec{x})$.

It will be convenient to consider a function $A\colon \mathbb{F}^m \to \mathbb{F}$ defined as $A(z) = \mathsf{DS}_q(z)$ for all $z \in \mathbb{F}^n$. Note that A agrees with q on at least α-fraction of points. Furthermore, note that for simplicity we may assume that q is the all zeros polynomial. Indeed, we can define $A'(x) := A(x) - q(x)$, and consider the case where the input is the all zeros polynomial, and the query algorithm is A'. Therefore, (1) we have $\Pr_{x \in \mathbb{F}^m}[A(x) = 0] \geq \alpha$, and (2) in the preprocessing phase we know that $q(\vec{w}) = 0$ for a random point $\vec{w}$, though it is not necessarily true that $A(\vec{w}) = 0$.

The following three claims complete the proof of Proposition 6.5.

Claim 6.6. *In the preprocessing phase, for a random choice of the reference point $\vec{w}$ with high probability over $\vec{x}$ it holds that $\vec{0} \in Q$, and hence in Q'. More formally, we have*

$$\mathbb{E}_{\vec{w} \in \mathbb{F}^m}[\Pr_x[\vec{0} \in Q]] \geq 1 - \frac{4}{|\mathbb{F}|\alpha} \ .$$

In particular, by Markov's inequality, for at least $1 - \sqrt{\frac{4}{|\mathbb{F}|\alpha}}$ *of* $\vec{w}$*'s it holds that*

$$\Pr_x[\vec{0} \in Q] \geq 1 - \sqrt{\frac{4}{|\mathbb{F}|\alpha}} \ . \tag{3}$$

Proof. It is a standard fact in the literature on derandomization (see, e.g., [MR06, Corollary 1.2]) that for any set $O \subseteq \mathbb{F}^n$ of size $|O| = \alpha|\mathbb{F}|^n$, and a random line $\ell_{\vec{x},\vec{w}} = \{\vec{x} + t(\vec{w} - \vec{x}) : t \in \mathbb{F}\}$ that passes through uniformly random $\vec{x}, \vec{w} \in \mathbb{F}^n$ it holds that

$$\Pr\left[\left|\frac{\left|\ell_{\vec{x},\vec{w}} \cap O\right|}{\left|\ell_{\vec{x},\vec{w}}\right|} - \alpha\right| > \varepsilon\right] \leq \frac{1}{|\mathbb{F}|}\frac{\alpha}{\varepsilon^2} \ .$$

The conclusion of the claim follows by letting $O = \{x \in \mathbb{F}^n : A(x) = 0\}$, and $\varepsilon = \alpha/2$. □

Claim 6.7 ([MR06, Proposition 3.5])**.** *Choose* $\vec{w}$ *and* $\vec{x}$ *uniformly at random and consider the line* $\ell_{\vec{x},\vec{w}}$*. Let* $Q = \{q_1, q_2, \ldots, q_k\}$ *be all the univariate polynomials of degree at most* d *that agree with* A *on at least* $\alpha/2$ *fraction of points in* $\ell_{\vec{x},\vec{w}}$*. If* $\alpha > 2\sqrt{\frac{d}{|\mathbb{F}|}}$*, then* $k \leq 2/\alpha$*.*

Claim 6.8. *Choose* $\vec{w}$ *and* $\vec{x}$ *uniformly at random and consider the line* $\ell_{\vec{x},\vec{w}}$*. Let* $Q = \{q_1, q_2, \ldots, q_k\}$ *be all the univariate polynomials of degree at most* d *that agree with* A *on at least* $\alpha/2$ *fraction of points in* $\ell_{\vec{x},\vec{w}}$*. Then, for all* $q_i \in Q$ *that are not identically zero it holds that* $\Pr[q_i(\vec{w}) = 0] \leq \frac{d}{|\mathbb{F}|}$*.*

In particular, if $\alpha > 2\sqrt{\frac{d}{|\mathbb{F}|}}$ *then*

$$\Pr_{\vec{w}}\left[\Pr_{\vec{x}}[\forall q_i \in Q \setminus \{\vec{0}\} : q_i(\vec{w}) \neq 0] \geq 1 - \sqrt{\frac{2d}{\alpha|\mathbb{F}|}}\right] \geq 1 - \sqrt{\frac{2d}{\alpha|\mathbb{F}|}} \ .$$

Proof. For any choice of $\vec{x}$ if $\vec{w}$ is chosen uniformly at random, and each univariate polynomial q_i is of degree d, the by Schwarz-Zippel lemma

$$\Pr[q_i(\vec{w}) = 0] \leq \frac{d}{|\mathbb{F}|} \ .$$

Also, by Claim 6.7, we know that $|Q| = k \leq 2/\alpha$, and hence, by union bound

$$\Pr_{\vec{x},\vec{w}}\left[\exists q_i \in Q \setminus \{\vec{0}\} : q_i(\vec{w}) = 0\right] \leq \frac{kd}{|\mathbb{F}|} \leq \frac{2d}{\alpha|\mathbb{F}|} \ .$$

This implies

$$\mathbb{E}_{\vec{w}}\left[\Pr_{\vec{x}}[\forall q_i \in Q \setminus \{\vec{0}\} : q_i(\vec{w}) \neq 0]\right] \geq 1 - \frac{kd}{|\mathbb{F}|} \geq 1 - \frac{2d}{\alpha|\mathbb{F}|} \ .$$

The claim follows by Markov's inequality. □

We now return to the proof of Proposition 6.5. By the claims above, for most $\vec{w}$'s it holds that if we choose $\vec{w}$ as a reference point, then for most $\vec{x}$'s, we have $\vec{0} \in Q$, and there is no other univariate

polynomial q_i in Q such that $q_i(\vec{w}) = 0$. More precisely, by combining Claim 6.6 with Claim 6.8 for at least $1 - O\left(\sqrt{\frac{d}{\alpha|\mathbb{F}|}}\right)$ fraction of $\vec{w}$'s it holds that

$$\Pr_x[\vec{0} \in Q \wedge \forall q \in Q : q \not\equiv 0, q(\vec{w}) \neq 0] \geq 1 - O\left(\sqrt{\frac{d}{\alpha|\mathbb{F}|}}\right) \ , \tag{4}$$

as required. □

Now we proceed with proving Lemma 6.3.

Proof of Lemma 6.3. Suppose there is a data structure DS as in the assumption of Lemma 6.3, with success probability $\Pr_{q,x}[\mathsf{DS}_q(x) = q(x)] \geq \alpha$. We show below how to construct a data structure DS' that will work for all input polynomial q and for most queries x.

Preprocessing:

Input: An input polynomial q of degree at most d

1. Identify q with a vector $q \in \mathbb{F}^n$ of its coefficients for $n = \binom{m+d}{d}$.
2. Let $Z = \{q \in \mathbb{F}^n : |X_q| \geq \frac{\alpha}{2} \cdot |\mathbb{F}|^n\}$.
3. Let O_Z be a membership oracle for Z that given a polynomial q' estimates $\frac{|X_{q'}|}{|\mathbb{F}|^n}$, the fraction of points on which $\mathsf{DS}_{q'}$ outputs $q'(x)$ correctly, within an additive error of $\alpha/10$, and returns ACCEPT if and only if the estimated fraction is more than $\alpha/3$.[a]
4. By applying Lemma 3.4, with probability $1 - \delta$ we obtain a vector u with at most $O(\log^4(1/\alpha))$ non-zero elements such that

   $$\Pr_{q_1,q_2,q_3 \in \mathbb{F}^n}[q_1, q_2, -q_3, -q_4 \in Z] \geq \Omega(\alpha^5) \ ,$$

   where $q_4 \in \mathbb{F}^n$ is such that $q - u = q_1 + q_2 - q_3 - q_4$.
5. Therefore, given q and s we can sample $O(\log(1/\delta) \cdot \log^4(1/\alpha))$ triplets of vectors until we find a triplet (q_1, q_2, q_3) and let $q_4 = q_1 + q_2 - q_3 - q - u$ satisfying $q_1, q_2 \in Z, -q_3, -q_4 \in Z$. Note that we can use the membership oracle O_Z to check that the vectors belong to Z.
6. Note that since each v_i belongs to Z, we have that DS outputs $p_i(x)$ correctly on at least $\alpha/4$ fraction of inputs. Thus, we can apply Proposition 6.5 on DS and obtain the data structure DS' such that $\Pr_{x \in \mathbb{F}^n}[\mathsf{DS}'_{q_i}(x) = q_i(x)] \geq 1 - O\left(\sqrt{\frac{d}{\alpha \cdot |\mathbb{F}|}}\right)$ for all $i = 1, 2$, and $\Pr_{x \in \mathbb{F}^n}[\mathsf{DS}'_{-q_j}(x) = -q_j(x)] \geq 1 - O\left(\sqrt{\frac{d}{\alpha \cdot |\mathbb{F}|}}\right)$ for $j = 3, 4$.
7. We store all the memory obtained by preprocessing the polynomials q_1, q_2, q_3, q_4 with DS. We also store the sparse vector s by storing the $O(\log^4(1/\alpha))$ non-zero coordinates, and their values.

[a] This is done by sampling $O(1/\alpha^2)$ uniformly random x's in $\mathbb{F}^n$, computing $\mathsf{DS}_{q'}(x)$ and $q'(x)$, and comparing the two results. In particular, if $|X_{q'}| \geq \frac{\alpha}{2} \cdot |\mathbb{F}|^n$, then $O_Z(q') = ACCEPT$ with probability $1 - \varepsilon$, and if $|X_{q'}| \leq \frac{\alpha}{4} \cdot |\mathbb{F}|^n$, then $O_Z(q') = REJECT$ with probability $1 - \varepsilon$.

For every polynomial q of degree at most d, let $X_q = \{x \in \mathbb{F}^m : \mathsf{DS}_q(x) = q(x)\}$. By averaging, there is a set Z of degree d polynomials such that $|Z| \geq \alpha/2 \cdot |\mathbb{F}|^n$, and $|X_q| \geq \alpha/2 \cdot |\mathbb{F}|^m$ for every $q \in Z$. Furthermore, note that it is straightforward to construct a membership oracle O_Z for Z, that given a polynomial q and access to DS_q estimates the fraction of queries x on which $\mathsf{DS}_q(x) = q(x)$.

Below we describe the preprocessing phase and the query phase of DS'.

The preprocessing on an input q works as follows. By identifying q with the vector of its coefficients in $\mathbb{F}^n$ with $n = \binom{m+d}{m}$, we use Lemma 3.4 to represent the vector q as $q = q_1 + q_2 - q_3 - q_4 + u$, where each $q_i \in Z$ and u is a sparse vector. Then, for each q_i we the reduction from Proposition 6.5 to obtain a data structure that works for each of the q_i for almost all queries x.

In the query phase, we use the data structures for each q_i to compute $q_i(x)$, and for the sparse polynomial s, we simply compute $u(x)$ using brute force. Finally, we return $q_1(x) + q_2(x) - q_3(x) - q_4(x) + u(x)$.

Preprocessing time and space: In the preprocessing step, we store the memory of the preprocessing for each of q_i and in addition the sparse vector u. Hence, the total space used is $4s + \log^4(1/\alpha)$ field elements + additional $\log^4(1/\alpha)$ coordinates of the input. Also, the running time is determined by number of samples needed to construct the oracle in Step 1 using Lemma 3.4. Both these steps take at most $O(\log^4(1/\alpha) \cdot \log(1/\delta))$ samples, bounding the running time of the preprocessing step.

Next we describe the query phase of our data structure.

Query phase:

Input: A query $\vec{x} \in \mathbb{F}^m$.
Recall that for the polynomial q, we have stored high-agreement data structures for evaluating q_1, q_2, q_3, q_3, together with a polynomial which is represented by a sparse vector of coefficient u.

1. For each polynomial q_i let $y_i = \mathsf{DS}'_{q_i}(\vec{x})$.
2. Compute $u(x)$. Since u has at most $O(\log^4(1/\alpha))$ non-zero coordinates, it follows that $u(x)$ can be computed in query time $O(\log^4(1/\alpha)\log(n))$.
3. Return $y_1 + y_2 - y_3 - y_4 + u(x)$.

Query time: The query time consists of querying the data structure 4 times, and evaluating $u(x)$. Note that the high-agreement data structure makes $|\mathbb{F}|$ queries to the weak-average-case data structure, and each query takes time t. Thus, the total query time is $4|\mathbb{F}| \cdot t + O(\log^4(1/\alpha)\log(n)))$.

Correctness: To prove the correctness, we bound the failure probability of the algorithm. Note that the algorithm returns the correct answer, unless for one of the polynomials q_i it holds that, $\mathsf{DS}'_{q_i}(\vec{x}) \neq q_i(\vec{x})$. This event happens with probability at most $O(\sqrt{\frac{d}{|\mathbb{F}|\alpha}})$. Hence, by applying union bound we can bound the failure probability to $O(\sqrt{\frac{d}{|\mathbb{F}|\alpha}})$. □

Finally, we prove Lemma 6.4.

Proof of Lemma 6.4. The proof of this lemma basically relies on the local decoding algorithm for Reed-Muller codes. Given a point $\vec{x}$, the query algorithm samples a random line $\ell_{x,y} = \{\vec{x}+r(\vec{y}-\vec{x}) : r \in \mathbb{F}\}$ passing through $\vec{x}$, and queries the data structure for all the points on this line. Given these values, the algorithm finds the closest univariate polynomial of degree at most d, call it h, and outputs $h(\vec{x})$. It is not hard to see that the algorithm succeeds with probability at least $1 - 4\gamma$.

For correctness since DS_p agrees with q on $1-\gamma$ fraction of the points $z \in \mathbb{F}^n$, it follows that for a random line ℓ through x the data structure DS satisfies $\Pr[agr \geq 3/4] \geq 1 - 4\gamma$, where agr denotes the fraction of points $z \in \ell$ with $\mathsf{DS}_{q_i}(z) = q_i(z)$. For each such line ℓ the only polynomial that agrees with DS on ℓ is q_i, and hence with probability at least $1-4\gamma$ the data structure outputs $q_i(x)$, as required. □

Putting it all together: Below we summarize the reductions above, and describe the full reduction that given a weak-average-case data structure DS that computes the correct answer for only α fraction of (q, x), gives us a data structure that works with high probability for all inputs q and all queries x.

We first use Lemma 6.4, reducing the problem to evaluating p on a random line ℓ passing through $\vec{x}$. Then, we apply Lemma 6.3, to write p as sum of 5 polynomials, for which we know one of them is sparse and can be computed efficiently, and the other four belong to the set of *good* polynomials, i.e., those polynomials for which the data structure succeeds in evaluating them on all but a small fraction of inputs. Finally, for each of these polynomials we apply the reduction in Proposition 6.5. This last step corresponds to choosing a random reference point $\vec{w} \in \mathbb{F}^n$, and passing lines between every $\vec{z} \in \ell$ and $\vec{w}$, and evaluating each of the q_i on each of the $\mathbb{F}$ lines.

References

[AD97] Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *STOC 1997*, pages 284–293, 1997.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems. In *STOC 1996*, pages 99–108, 1996.

[Aka08] Adi Akavia. Learning significant fourier coefficients over finite abelian groups. In Ming-Yang Kao, editor, *Encyclopedia of Algorithms - 2008 Edition*. Springer, 2008.

[AMN98] Yossi Azar, Rajeev Motwani, and Joseph Seffi Naor. Approximating probability distributions using small sample spaces. *Combinatorica*, 18(2):151–171, 1998.

[AV21] Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In *SODA 2021*, pages 522–539. SIAM, 2021.

[BABB19] Enric Boix-Adserà, Matthew Brennan, and Guy Bresler. The average-case complexity of counting cliques in Erdős–Rényi hypergraphs. In *FOCS 2019*, pages 1256–1280. IEEE, 2019.

[BFNW93] László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3(4):307–318, 1993.

[BLR90] Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. In *STOC 1990*, pages 73–83. ACM, 1990.

[BRSV17] Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. Average-case fine-grained hardness. In *STOC 2017*, pages 483–496, 2017.

[BRSV18] Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. Proofs of work from worst-case assumptions. In *CRYPTO 2018*, pages 789–819. Springer, 2018.

[BRTW14] Eli Ben-Sasson, Noga Ron-Zewi, Madhur Tulsiani, and Julia Wolf. Sampling-based proofs of almost-periodicity results and algorithmic applications. In *ICALP 2014*, pages 955–966. Springer, 2014.

[BT06] Andrej Bogdanov and Luca Trevisan. Average-case complexity. *Foundations and Trends in Theoretical Computer Science*, 2(1):1–106, 2006.

[CGL15] Raphael Clifford, Allan Grønlund, and Kasper Green Larsen. New unconditional hardness results for dynamic and online problems. In *FOCS 2015*, pages 1089–1107. IEEE, 2015.

[Cha02] Mei-Chu Chang. A polynomial bound in Freiman's theorem. *Duke Mathematical Journal*, 113(3):399 – 419, 2002.

[CKL18] Diptarka Chakraborty, Lior Kamma, and Kasper Green Larsen. Tight cell probe bounds for succinct boolean matrix-vector multiplication. In *STOC 2018*, pages 1297–1306, 2018.

[CKLM18] Arkadev Chattopadhyay, Michal Kouckỳ, Bruno Loff, and Sagnik Mukhopadhyay. Simulation beats richness: New data-structure lower bounds. In *STOC 2018*, pages 1013–1020, 2018.

[CS10] Ernie Croot and Olof Sisask. A probabilistic technique for finding almost-periods of convolutions. *Geometric and Functional Analysis*, 20(6):1367—-1396, 2010.

[DKKS21] Pavel Dvořák, Michal Kouckỳ, Karel Král, and Veronika Slívová. Data structures lower bounds and popular conjectures. *arXiv:2102.09294*, 2021.

[DLV20] Mina Dalirrooyfard, Andrea Lincoln, and Virginia Vassilevska Williams. New techniques for proving fine-grained average-case hardness. In *FOCS 2020*, pages 774–785. IEEE, 2020.

[FF93] Joan Feigenbaum and Lance Fortnow. Random-self-reducibility of complete sets. *SIAM Journal on Computing*, 22(5):994–1005, 1993.

[FHM01] Gudmund Skovbjerg Frandsen, Johan P. Hansen, and Peter Bro Miltersen. Lower bounds for dynamic algebraic problems. *Information and Computation*, 171(2):333–349, 2001.

[Fre77] Rusins Freivalds. Probabilistic machines can use less running time. In *IFIP 1977*, pages 839–842, 1977.

[GR18] Oded Goldreich and Guy Rothblum. Counting t-cliques: Worst-case to average-case reductions and direct interactive proof systems. In *FOCS 2018*, pages 77–88. IEEE, 2018.

[HKNS15] Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *STOC 2015*, pages 21–30, 2015.

[HLS21] Monika Henzinger, Andrea Lincoln, and Barna Saha. The complexity of average-case dynamic subgraph counting. *ECCC*, 2021.

[Imp95] Russell Impagliazzo. A personal view of average-case complexity. In *CCC 1995*, pages 134–147. IEEE, 1995.

[Imp11] Russell Impagliazzo. Relativized separations of worst-case and average-case complexities for NP. In *CCC 2011*, pages 104–114. IEEE, 2011.

[Juk12] Stasys Jukna. *Boolean function complexity: advances and frontiers*, volume 27. Springer Science & Business Media, 2012.

[KU08] Kiran S. Kedlaya and Christopher Umans. Fast modular composition in any characteristic. In *FOCS 2008*, pages 146–155. IEEE, 2008.

[Lar12] Kasper Green Larsen. Higher cell probe lower bounds for evaluating polynomials. In *FOCS 2012*, pages 293–301. IEEE, 2012.

[Lev86] Leonid A. Levin. Average case complete problems. *SIAM Journal on Computing*, 15(1):285–286, 1986.

[Lip91] Richard Lipton. New directions in testing. *Distributed computing and cryptography*, 2:191–202, 1991.

[LLV19] Rio LaVigne, Andrea Lincoln, and Virginia Vassilevska Williams. Public-key cryptography in the fine-grained setting. In *CRYPTO 2019*, pages 605–635. Springer, 2019.

[Lov15] Shachar Lovett. An exposition of Sanders' quasi-polynomial Freiman-Ruzsa theorem. *Theory of Computing*, pages 1–14, 2015.

[Lov17] Shachar Lovett. Additive combinatorics and its applications in theoretical computer science. *Theory of Computing*, pages 1–55, 2017.

[LW17] Kasper Green Larsen and Ryan Williams. Faster online matrix-vector multiplication. In *SODA 2017*, pages 2182–2189. SIAM, 2017.

[MR06] Dana Moshkovitz and Ran Raz. Sub-constant error low degree test of almost-linear size. In *STOC 2006*, pages 21–30. ACM, 2006.

[NN93] Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM J. Comput.*, 22(4):838–856, 1993.

[Reg04] Oded Regev. New lattice-based cryptographic constructions. *Journal of the ACM*, 51(6):899–942, 2004.

[San12] Tom Sanders. On the Bogolyubov–Ruzsa lemma. *IEEE Trans. Inf. Theory*, 5(3):627–655, 2012.

[Sho09] Victor Shoup. *A computational introduction to number theory and algebra*. Cambridge, 2009.

[STV01] Madhu Sudan, Luca Trevisan, and Salil Vadhan. Pseudorandom generators without the XOR lemma. *Journal of Computer and System Sciences*, 62(2):236–266, 2001.

[TS17] Amnon Ta-Shma. Explicit, almost optimal, epsilon-balanced codes. In *STOC 2017*, pages 238–251, 2017.

[Vas18] Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *ICM 2018*, 2018.

A Proof of the probabilistic version of Sanders' lemma

Below we prove Lemma 3.3. The proof follows the approach of Sanders, with several modifications. We follow the exposition of Lovett [Lov15] in the proof of the lemma.

Lemma 3.3 (Probabilistic quasi-polynomial Bogolyubov-Ruzsa lemma)**.** *Let $\mathbb{F} = \mathbb{F}_p$ be a prime field, and let $A \subseteq \mathbb{F}^n$ be a set of size $|A| = \alpha \cdot |\mathbb{F}|^n$, for some $\alpha \in (0,1]$. Then, there exists a subspace $V \subseteq \mathbb{F}^n$ of dimension $\dim(V) \geq n - O(\log^4(1/\alpha))$ such that for all $v \in V$ it holds that*

$$\Pr_{a_1,a_2,a_3 \in \mathbb{F}^n}[a_1, a_2 \in A, a_3, a_4 \in -A] \geq \Omega(\alpha^5) \ ,$$

where $a_4 = v - a_1 - a_2 - a_3$. Furthermore, given a query access to the set A, there is an algorithm that runs in time $\exp(\log^4(1/\alpha)) \cdot \operatorname{poly}\log(1/\delta) \cdot \operatorname{poly}(n)$ and with probability $1 - \delta$ computes a set of vectors $R \subseteq \mathbb{F}^n$ such that $V = \{v \in \mathbb{F}^n : \langle v, r\rangle = 0 \ \forall r \in R\}$.

Before starting with the proof, let us establish some notation. For a set $A \subseteq \mathbb{F}^n$, we denote by $1_A \colon \mathbb{F}^n \to \{0,1\}$ the indicator function of A, where $1_A(x) = 1$ if $x \in A$, and $1_A(x) = 0$ otherwise. In particular $\mathbb{E}_{x \in \mathbb{F}^n}[1_A(x)] = \frac{|A|}{|\mathbb{F}|^n}$. We also let $\varphi_A \colon \mathbb{F}^n \to \mathbb{R}$ be the normalization of 1_A defined as $\varphi_A(x) = 1_A(x) \cdot \frac{|\mathbb{F}|^n}{|A|}$ so that $\mathbb{E}_{x \in \mathbb{F}^n}[\varphi_A(x)] = 1$. When the set A is a singleton $A = \{a\}$, we will write $\varphi_a = \varphi_{\{a\}}$.

It is easy to verify that for a set A and a function f the convolution $\varphi_A * f$ is given by $\varphi_A * f(x) = \mathbb{E}_{a \in A}[f(x-a)]$. In particular, for $a \in \mathbb{F}^n$ we have $\varphi_a * f(x) = f(x-a)$.

As a starting point, we define the set $D = \{d \in \mathbb{F}^n : 1_A * 1_{-A}(d) \geq \delta\}$ for a parameter $\delta = \alpha^2/20$. That is, D is the set of all *popular differences* of two elements of A. In other words, D consists of all $d \in \mathbb{F}^n$ such that there are $\delta|\mathbb{F}|^n$ pairs $(a, a') \in A^2$ satisfying $d = a - a'$, i.e., $\Pr_{a \in \mathbb{F}^n, a' = d-a}[a \in A, a' \in -A] \geq \delta$.

Note first that $\langle 1_{A-A}, \varphi_A * \varphi_{-A}\rangle = \mathbb{E}_{x,y \in \mathbb{F}^n}[1_{A-A}(x-y)\varphi_A(x)\varphi_{-A}(-y)] = \mathbb{E}_{x,y \in A}[1_{A-A}(x-y)] = 1$. Next we observe that D approximates $A - A$, in the sense that $\langle 1_D, \varphi_A * \varphi_{-A}\rangle \geq 1 - \frac{\delta}{\alpha^2} = 0.95$. Indeed, using the fact that $D \subseteq A - A$, we have

$$\begin{aligned}
\langle 1_D, \varphi_A * \varphi_{-A}\rangle &= \langle 1_{A-A}, \varphi_A * \varphi_{-A}\rangle - \left\langle 1_{\mathbb{F}^n \setminus D}, \varphi_A * \varphi_{-A}\right\rangle \\
&= 1 - \frac{1}{\alpha^2}\left\langle 1_{\mathbb{F}^n \setminus D}, 1_A * 1_{-A}\right\rangle \\
&= 1 - \frac{1}{\alpha^2} \cdot \Pr_{d,a \in \mathbb{F}^n}[a \in A, d - a \in -A | d \notin D] \cdot \Pr_{d \in \mathbb{F}^n}[d \notin D] \\
&\geq 1 - \frac{\delta}{\alpha^2} \, . \qquad (5)
\end{aligned}$$

We remark that this is one of the main differences in our proof compared to the original proof of Sanders, who only relied on the fact that $\langle 1_{A-A}, \varphi_A * \varphi_{-A}\rangle = 1$.

The proof of Lemma 3.3 consists of the following two parts.

Lemma A.1. *Let $A \subseteq \mathbb{F}^n$ be a set of size $|A| = \alpha|\mathbb{F}|^n$. Set $t = O(\log(1/\alpha))$. There exists a set $X \subseteq \mathbb{F}^n$ of size $|X| \geq \alpha^{O(\log^3(1/\alpha))}|\mathbb{F}|^n$ such that for all $x_1, x_2, \ldots, x_t \in X$ it holds that*

$$\Pr_{a_1,a_2 \in A}[a_1 - a_2 - \sum_{i=1}^{t} x_i \in D] \geq 0.9 \ . \qquad (6)$$

Given the set X from Lemma A.1 we use a standard Fourier-analytic argument to define a large subspace V such that $|D \cap V'| \geq 0.8|V|$ where V' is some coset of V. In fact, we show that there are many such cosets. Formally, we prove the following lemma.

Lemma A.2. *Let $A \subseteq \mathbb{F}^n$ be a set of size $|A| = \alpha|\mathbb{F}|^n$, Then, there exists a subspace $V \subseteq \mathbb{F}^n$ of dimension $\dim(V) = n - O(\log^4(1/\alpha))$ and a vector $b \in \mathbb{F}^n$ such that if we sample a uniformly random $a \in A$ and $v \in V$ then*

$$\Pr_{a \in A, v \in V}[v + a + b \in D] \geq 0.85 \ . \tag{7}$$

Remark A.3. *Lemma A.1 corresponds to Lemma 4.2 in [Lov15]. The only difference is that we claim that the sum belongs to D with high probability, while in [Lov15] the sum belongs to $A - A$. Note that the two statements are indeed close to each other by Eq.* (5).

Lemma A.2 roughly corresponds to the conclusion of the section "A Fourier-analytic argument" *in [Lov15].*

We show next how to conclude the proof of Lemma 3.3 from Lemma A.2. Indeed, by Eq. (7) it follows that $\mathbb{E}_{a \in A}[\Pr_{v \in V}[v + a + b \in D]] \geq 0.85$, and hence for at least $0.05|A|$ many $a \in A$ it holds that $\Pr_{v \in V}[v + a + b \in D] \geq 0.8$. In particular, for $C = \{c \in \mathbb{F}^n : |D \cap (V + c)| \geq 0.8|V|\}$ we have $|C| \geq 0.05\alpha|\mathbb{F}|^n$.

Claim A.4. *The set $C \subseteq \mathbb{F}^n$ is symmetric. Namely, if $c \in C$, then $-c \in C$.*

Proof. Let $c \in C$ and let $V + c$ be the corresponding coset of V. We claim that $-c \in C$, i.e., $\left|\{v \in V : \text{ there are } \geq \delta|\mathbb{F}|^n \text{ pairs } (a_1, a_2) \in A^2 \text{ such that } a_1 - a_2 = v - c\}\right| \geq 0.8|V|$.

For $c \in C$ let $P_c = \{v \in V : \text{there are } \geq \delta|\mathbb{F}|^n \text{ pairs } (a_1, a_2) \in A^2 \text{ such that } a_1 - a_2 = -v + c\}$. By definition of C, we have $|P_c| \geq 0.8|V|$.

To see that $-c \in C$ take any $v \in P_C$, and note that $a_1 - a_2 = -v + c$ if and only if $a_2 - a_1 = v - c$. Therefore, for each $v \in P_c$ there are $\geq \delta|\mathbb{F}|^n$ pairs $(a_1, a_2) \in A^2$ such that $a_2 - a_1 = v - c$, and thus $-c \in C$, as required. □

Let us choose a unique representative c^* for each coset $V + c$ of V such that $|D \cap (V + c)| \geq 0.8|V|$, and let $C^* = \{c^* \text{ is the representative of } V + c : |D \cap (V + c)| \geq 0.8|V|\}$. And furthermore, let us assume without loss of generality that C^* is symmetric, i.e. $c^* \in C^*$ implies that $-c^* \in C^*$. Then, the union of all these coset covers is at least 0.05α fraction of $\mathbb{F}^n$, i.e.,

$$|\cup_{c^* \in C^*}(V + c^*)| \geq 0.05\alpha|\mathbb{F}|^n \ . \tag{8}$$

We are now ready to show that

$$\Pr_{\substack{a_1, a_2, a_3 \in \mathbb{F}^n \\ a_4 = v - a_1 - a_2 - a_3}} [a_1, a_2 \in A, a_3, a_4 \in -A] \geq \Omega(\alpha^5) \ .$$

Proof of Lemma 3.3. Fix $v \in V$. Since $|D \cap (V + c^*)| \geq 0.8|V|$ for every coset $V + c^*$ such that $c^* \in C^*$, it follows by the symmetry of C^* that for every $c^* \in C^*$ we have at least $0.1 \cdot |V|$ pairs $(u + c^*, v - u - c^*) \in D^2$ such that $u \in V$. Therefore, by Eq. (8) for every $v \in V$ there are at least $0.1 \times 0.05\alpha|\mathbb{F}|^n$ different pairs $(u + c^*, v - u - c^*)$ such that both $u + c^* \in D \cap (V + c^*)$ and $v - u - c^* \in D \cap (V - c^*)$.

Letting $d_1 = u + c^*$, so far we got that for every $v \in V$ we have $\Pr_{d_1 \in \mathbb{F}^n, d_2 = v - d_1}[d_1, d_2 \in D] \geq \Omega(\alpha)$.

Recall that by the definition of D, every $d_1 \in D$ is a popular difference of elements of A, i.e. $\Pr_{\substack{a_1 \in \mathbb{F}^n \\ a_3 = d_1 - a_1}}[a_1 \in A, a_3 \in -A] \geq \delta$. Similarly, for $d_2 \in D$ we have $\Pr_{\substack{a_2 \in \mathbb{F}^n \\ a_4 = d_2 - a_2}}[a_2 \in A, a_4 \in -A] \geq \delta$. This implies that

$$\Pr_{\substack{a_1, a_2, a_3 \in \mathbb{F}^n \\ a_4 = v - a_1 - a_2 - a_3}}[a_1, a_2 \in A, a_3, a_4 \in -A] \geq \Omega(\alpha \cdot \delta^2) \ ,$$

as required. □

We now turn to proving each of the two steps stated in Lemma A.1 and Lemma A.2.

A.1 Proof of Lemma A.1

The proof starts with the following lemma of Croot and Sisask [CS10].

Lemma A.5 (Croot-Sisask [CS10, Proposition 3.3])**.** *Let $A, B \subseteq \mathbb{F}^n$ be two sets, and let $\varepsilon \in (0,1)$ and $p \geq 1$. Let $\alpha = \frac{|A|}{|\mathbb{F}|^n} \in (0,1)$. Then, there exists a set $X \subseteq \mathbb{F}^n$ of size $|X| \geq (\alpha/2)^{O(p/\varepsilon^2)}|\mathbb{F}|^n$ such that for all $x \in X$ it holds that*

$$\|\varphi_x * \varphi_A * 1_B - \varphi_A * 1_B\|_p \leq \varepsilon \, .$$

Let $p = \log_2(1/\alpha)$, $t = \Theta(\log(1/\alpha))$, and $\varepsilon = (1/40t)$. By applying Lemma A.5 we obtain a set $X \subseteq \mathbb{F}^n$ of size $|X| \geq (\alpha/2)^{O(p/\varepsilon^2)} \geq \alpha^{O(\log^3(1/\alpha))}|\mathbb{F}|^n$. We show below that X satisfies Eq. (6).

Fix $x_1, \ldots, x_t \in X$, and let $s = \sum_{i=1}^t x_i$. Note first that by setting $B = D$ in Lemma A.5 and combining it with triangle inequality we get that

$$\|\varphi_s * \varphi_A * 1_D - \varphi_A * 1_D\|_p \leq t \cdot \varepsilon \leq 1/40 \ ,$$

where the last inequality is by the choice of $\varepsilon = 1/40t$. Let $q = p/(p-1)$, then by the choice of $p = \log_2(1/\alpha)$ we have

$$\|\varphi_A\|_q = \left(\alpha \cdot \frac{1}{\alpha^q}\right)^{1/q} = \left(\frac{1}{\alpha}\right)^{1/p} \leq 2 \ .$$

Then, by applying Hölder's inequality with $q = p/(p-1)$ we get

$$|\langle \varphi_s * \varphi_A * 1_D - \varphi_A * 1_D, \varphi_A \rangle| \leq \|\varphi_s * \varphi_A * 1_D - \varphi_A * 1_D\|_p \cdot \|\varphi_A\|_q \leq 1/20 \ .$$

By combining the above inequality with Eq. (5) we get

$$\begin{aligned}
\Pr_{a_1, a_2 \in A}[a_1 - a_2 - s \in D] &= \langle \varphi_s * \varphi_A * 1_D, \varphi_A \rangle \\
&= \langle \varphi_A * 1_D, \varphi_A \rangle - \langle \varphi_A * 1_D - \varphi_s * \varphi_A * 1_D, \varphi_A \rangle \\
&= \langle 1_D, \varphi_A * \varphi_{-A} \rangle - \langle \varphi_A * 1_D - \varphi_s * \varphi_A * 1_D, \varphi_A \rangle \\
&\geq (1 - \delta/\alpha^2) - 1/20 \geq 0.9 \ ,
\end{aligned}$$

as required.

A.2 Proof of Lemma A.2

The proof of this step is essentially Section 5 of [Lov15]. The only (minor) difference is that we work over $\mathbb{F}_p$ and not over $\mathbb{F}_2$.

Given the set $X \subseteq \mathbb{F}^n$ from Lemma A.1 of size $|X| \geq \alpha^{O(\log^3(1/\alpha))}|\mathbb{F}|^n$, we define $\mathrm{Spec}_\gamma(X) = \{r \in \mathbb{F}^n : |\widehat{\varphi}_X(r)| \geq \gamma\}$. Since $\sum_{q\in\mathbb{F}^n} |\widehat{\varphi}_X(r)|^2 = \mathbb{E}_z[\varphi_X(z)^2] = 1/\alpha$, it follows that $\left|\mathrm{Spec}_\gamma(X)\right| \leq \frac{1}{\alpha\gamma^2}$. Chang's lemma provides a non-trivial bound on the dimension of the subspace containing $\mathrm{Spec}_\gamma(X)$.

Lemma A.6 (Chang [Cha02])**.** *Let $X \subseteq \mathbb{F}^n$ of size $|X| = \beta \cdot |\mathbb{F}|^n$, and let $\gamma > 0$. Then*

$$\dim(\mathrm{Spec}_\gamma(X)) \leq O\left(\frac{\log(1/\beta)}{\gamma^2}\right) \ .$$

Define the subspace $V = \mathrm{Spec}_{1/2}(X)^\perp = \{v \in \mathbb{F}^n : \langle v, r\rangle = 0 \ \forall r \in \mathrm{Spec}_{1/2}(X)\}$. Lemma A.6 implies that $\dim(V) \geq n - O(\log^4(|\mathbb{F}|^n/|X|)) \geq n - O(\log^4(1/\alpha))$.

We now show that V indeed satisfies the guarantee of Lemma A.2. For $x_1, x_2, \ldots, x_t \in X$ let $s = \sum_{i=1}^t x_i$ as in the previous lemma. By Lemma A.1 for all $x_1, x_2, \ldots, x_t \in X$ we have

$$\Pr_{a_1,a_2\in A}[a_1 - a_2 - s \in D] \geq 0.9 \ .$$

Next, we are comparing this probability to the following.

$$\Pr_{\substack{v\in V\\ a_1,a_2\in A}}[v + a_1 - a_2 - s \in D] \ . \tag{9}$$

We claim that if we sample $v \in V$, $a_1, a_2 \in A$, and $x_1, \ldots, x_t \in X$ uniformly at random (and let $s = \sum_{i=1}^t x_i$), then the two quantities are close to each other. We prove this by rewriting the two probabilities using the Fourier expansion. Note that the Fourier coefficients of φ_V are simple to describe since V is a linear subspace, and they are equal to $\widehat{\varphi_V}(r) = 1$ if $r \in V^\perp$ and $\widehat{\varphi_V}(r) = 0$ otherwise. Therefore,

$$\begin{aligned}
\Pr_{\substack{v\in V\\ x_1,\ldots,x_t\in X\\ a_1,a_2\in A}}[v + a_1 - a_2 - s \in D] &= \left\langle \varphi_V * \varphi_A * \varphi_{-A} * \varphi_{-X}^{(t)}, \mathbf{1}_D \right\rangle \\
&= \sum_{r\in\mathbb{F}^n} \widehat{\varphi_V}(r) \cdot \widehat{\varphi_A}(r) \cdot \widehat{\varphi_A}(-r) \cdot \widehat{\varphi_X}^t(-r) \cdot \overline{\widehat{\mathbf{1}_D}(r)} \\
&= \sum_{r\in V^\perp} \widehat{\varphi_A}(r) \cdot \widehat{\varphi_A}(-r) \cdot \widehat{\varphi_X}^t(-r) \cdot \overline{\widehat{\mathbf{1}_D}(r)} \ .
\end{aligned}$$

On the other hand

$$\Pr_{\substack{x_1,\ldots,x_t\in X\\ a_1,a_2\in A}}[a_1 - a_2 - s \in D] = \left\langle \varphi_A * \varphi_{-A} * \varphi_{-X}^{(t)}, \mathbf{1} \right\rangle = \sum_{r\in\mathbb{F}^n} \widehat{\varphi_A}(r) \cdot \widehat{\varphi_A}(-r) \cdot \widehat{\varphi_X}^t(-r) \cdot \overline{\widehat{\mathbf{1}_D}(r)} \ .$$

This implies

$$\begin{aligned}
\left| \Pr_{\substack{v\in V\\ x_1,\ldots,x_t\in X\\ a_1,a_2\in A}}[v+a_1-a_2-s\in D] - \Pr_{\substack{x_1,\ldots,x_t\in X\\ a_1,a_2\in A}}[a_1-a_2-s\in D] \right| &= \sum_{r\notin V^\perp} \widehat{\varphi_A}(r)\cdot\widehat{\varphi_A}(-r)\cdot\widehat{\varphi_X}^t(r)\cdot\overline{\widehat{\mathbf{1}_D}(r)} \\
&\leq \sum_{r\notin V^\perp} \left|\widehat{\varphi_A}(r)\cdot\widehat{\varphi_A}(-r)\cdot 2^{-t}\cdot\overline{\widehat{\mathbf{1}_D}(r)}\right| \\
&\leq 2^{-t}\sum_{r\in\mathbb{F}^n} |\widehat{\varphi_A}(r)\cdot\widehat{\varphi_A}(-r)| \\
\text{By Cauchy-Schwarz} \quad &\leq 2^{-t}\sum_{r\in\mathbb{F}^n} \left|\widehat{\varphi_A}^2(r)\right| \\
&= 2^{-t}\cdot\mathbb{E}_{x\in\mathbb{F}^n}[\varphi_A^2(x)] \\
&= \frac{1}{\alpha\cdot 2^t} < 0.05 \ ,
\end{aligned}$$

where the last inequality holds due to the choice of $t = O(\log(1/\alpha))$. Therefore,

$$\Pr_{\substack{v\in V\\ x_1,\ldots,x_t\in X\\ a_1,a_2\in A}}[v+a_1-a_2-s\in D] \geq \Pr_{\substack{x_1,\ldots,x_t\in X\\ a_1,a_2\in A}}[a_1-a_2-s\in D] - 0.05 \geq 0.85 \ .$$

Finally, we can fix a_2+s maximizing the probability, and let $b=-a_2-s$ to conclude the proof of Lemma A.2.

A.3 Algorithmic construction of the subspace V

Given a set A we can construct V using the algorithm described in [BRTW14]. Indeed, the only ingredients we need for our construction are the set X from Lemma A.1 and the subspace V guaranteed by Lemma A.2.

A straightforward inspection of the algorithm described in [BRTW14] gives the desired result. Informally, the algorithm works as follows: The set X is defined as the set of all $x\in X$ such that

$$\|\varphi_x * \varphi_A * 1_D - \varphi_A * 1_D\|_p \leq \varepsilon \ .$$

Note that given $x\in\mathbb{F}^n$ we can estimate the norm efficiently (up to a small error). This gives us a membership oracle to the set X.

Given such a query oracle, we can use Goldreich-Levin algorithm over $\mathbb{F}$ to compute $R = \mathrm{Spec}_{1/2}(X)$ [Aka08], or more precisely its superset that is not too large, and using it we define the subspace $V = \{v\in\mathbb{F}^n : \langle v,r\rangle = 0\ \forall r\in R\}$.