Electronic Colloquium on Computational Complexity, Report No. 39 (2017)

Average-Case Fine-Grained Hardness

Marshall Ball* Alon Rosen† Manuel Sabin‡ Prashant Nalini Vasudevan§

February 27, 2017

Abstract

We present functions that can be computed in some fixed polynomial time but are hard *on average* for any algorithm that runs in slightly smaller time, assuming widely-conjectured *worst-case* hardness for problems from the study of fine-grained complexity. Unconditional constructions of such functions are known from before (Goldmann et al., IPL '94), but these have been canonical functions that have not found further use, while our functions are closely related to well-studied problems and have considerable algebraic structure.

We prove our hardness results in each case by showing fine-grained reductions from solving one of three problems – namely, Orthogonal Vectors (OV), 3SUM, and All-Pairs Shortest Paths (APSP) – in the worst case to computing our function correctly on a uniformly random input. The conjectured hardness of OV and 3SUM then gives us functions that require $n^{2-o(1)}$ time to compute on average, and that of APSP gives us a function that requires $n^{3-o(1)}$ time. Using the same techniques we also obtain a conditional average-case time hierarchy of functions.

Based on the average-case hardness and structural properties of our functions, we outline the construction of a *Proof of Work* scheme and discuss possible approaches to constructing fine-grained One-Way Functions. We also show how our reductions make conjectures regarding the worst-case hardness of the problems we reduce from (and consequently the Strong Exponential Time Hypothesis) *heuristically falsifiable* in a sense similar to that of (Naor, CRYPTO '03).

Keywords: Average-Case Complexity, Fine-Grained Complexity, Cryptography, Heuristic Falsifiability, Proofs of Work, Worst-Case to Average-Case Reduction.

*Columbia University, New York, NY, USA. Email: marshall@cs.columbia.edu.

†Efi Arazi School of Computer Science, IDC Herzliya, Israel. Email: alon.rosen@idc.ac.il.

‡UC Berkeley, Berkeley, CA, USA. Email: msabin@berkeley.edu.

§CSAIL, Massachusetts Institute of Technology, Cambridge, MA, USA. Email: prashvas@mit.edu.

ISSN 1433-8092

1 Introduction

Since the 1970s we have had a notion of what we consider "easy" and what we consider "hard." Polynomial-time computable has long been synonymous to efficient and easy, while showing a problem NP-complete was to condemn it as intractable. In our recent history, however, this categorization has been called into question: SAT instances, the flagship of NP-complete problems, are solved on the daily [BHvM09], while algorithms that run in as little as quadratic time may be prohibitively expensive for some practical problems such as DNA sequencing, due to large input sizes.

Thus, in the "real world," our notions of easy and hard may not always align with our classical views. The main problem here is our choice of analysis. For SAT, we classify it as "hard" when it often may be more appropriately classified as "easy" because complexity theory typically employs *worst-case* analysis. That is, we may be adhering to an overly-pessimistc metric, when, in practice, the SAT instances we come across may be much more benign. In part to combat this sort of problem, average-case complexity was introduced in [Lev86]. By considering distributions over problem instances, we can at least hope to argue about the performance of heuristic algorithms in practice.

Similarly, the *practical hardness* of a problem with quadratic time complexity is invisible to our typical "coarse-grained" analysis that only distinguishes between polynomial and not polynomial. Within the past decade, the field of fine-grained complexity has quickly developed [Wil15], mapping out (conditional) hardness of natural problems *within* P. By introducing fine-grained reductions, a picture is emerging of a few main islands amongst the web of reductions, giving us an increasingly clearer classification of the relative hardness of fine-grained problems. Through such reductions, the more exact practical hardness of problems, such as DNA sequencing's quadratic time barrier [BI14], has been given evidence for.

However, while average-case analysis and fine-grained analysis independently address issues in classical complexity theory, average-case analysis is still coarse-grained and fine-grained analysis is still worst-case. A more complete theory attempting to capture the notion of "complexity" in our world should begin by marrying average-case and fine-grained analysis.

In this paper we do so by providing average-case fine-grained hardness conjectures and show them to follow from widely conjectured worst-case assumptions on well-studied fine-grained problems. Alternatively viewed, we give new routes for the falsifications of these worst-case conjectures.

1.1 Our Results

We present worst-case–to–average-case fine-grained reductions from the three main islands of fine-grained complexity theory. We recall these three problems here to frame our work, and their relevance is discussed in Section 2.

- *Orthogonal Vectors:* The OV problem on vectors of dimension d (denoted OV_d) is to determine, given two sets U, V of n vectors from $\{0,1\}^{d(n)}$ each, whether there exist $u \in U$ and $v \in V$ such that $\langle u, v\rangle = 0$ (over $\mathbb{Z}$). (If left unspecified, d is to be taken to be $\lceil \log^2 n \rceil$.)
- *3SUM:* The 3SUM problem is to determine whether a given set $S \subset \{-n^3, \ldots, n^3\}$ of size n contains three distinct elements a, b, c such that $a + b = c$.
- *All-Pairs Shortest Path*: Given an edge-weighted (undirected or directed) graph on n vertices, the APSP problem is to find the distances between every pair of vertices, where edge weights are in $\{1, \ldots, n^c\}$ for some sufficiently large c.

We give a family of polynomials over finite fields corresponding to each of these, called $\mathcal{F}$OV, $\mathcal{F}$3SUM, and $\mathcal{F}$ZWT respectively, and conjecture these polynomials to be hard to evaluate on

uniformly chosen inputs. To support these conjectures we prove worst-case–to–average-case fine-grained reductions from OV, 3SUM, and APSP to their respective families of polynomials (where 3SUM reduces also to $\mathcal{F}$ZWT). Specifically, we show:

- If OV requires $n^{2-o(1)}$ time to decide in the worst-case, then $\mathcal{F}$OV requires $n^{2-o(1)}$ time to evaluate with probability 3/4 on uniformly chosen inputs.
- If 3SUM requires $n^{2-o(1)}$ time to decide in the worst-case, then $\mathcal{F}$3SUM requires $n^{2-o(1)}$ time to evaluate with probability 3/4 on uniformly chosen inputs.
- If APSP requires $n^{3-o(1)}$ time *or* 3SUM requires $n^{2-o(1)}$ time to decide in the worst-case, then $\mathcal{F}$ZWT requires $n^{3-o(1)}$ time to evaluate with probability 3/4 on uniformly chosen inputs.

Further, we conjecture a fourth family of polynomials, $\mathcal{F}$TC, to also be average-case hard and support this with fine-grained reductions from 3SUM, and APSP, and k-SAT. The reduction from k-SAT makes $\mathcal{F}$TC hard under the *Strong Exponential Time Hypothesis* (SETH), which states that there is no $\epsilon > 0$ such that k-SAT can be solved in time $\widetilde{O}(2^{n(1-\epsilon)})$ for all values of k.

- If *either* APSP requires $n^{3-o(1)}$ time, 3SUM requires $n^{2-o(1)}$ time, *or* SETH holds, then $\mathcal{F}$TC requires $n^{3-o(1)}$ time to evaluate with probability 3/4 on uniformly chosen inputs.

We note that SETH implies that OV requires $n^{2-o(1)}$ time to decide in the worst-case and so $\mathcal{F}$OV is also hard on average under the stronger assumption of SETH. Thus, $\mathcal{F}$OV and $\mathcal{F}$TC are our mostly strongly supported average-case hardness results. $\mathcal{F}$TC only becomes easy if SETH breaks and both 3SUM and APSP, while $\mathcal{F}$OV, even with a broken SETH, remains hard unless *all* first-order graph problems become easy since [GI16] shows that all such problems reduce to OV.[1]

Our results crucially rely on the fact that the polynomials in $\mathcal{F}$OV, $\mathcal{F}$3SUM, $\mathcal{F}$ZWT, and $\mathcal{F}$TC have degree polylog(n), which is very low. This extremely low degree enables us to invoke *in a fine-grained way* the classic random self-reducibility of evaluating low-degree polynomials, first used to show the average-case hardness of computing the Permanent when assuming its worst-case hardness [Lip91, FF90], or more generally to show local correctability of Reed-Muller codes [GS92].

Beyond low degree, our polynomials are efficient to evaluate in time that *tightly matches* their conjectured average-case hardness. These two properties are what distinguishes our polynomials from naïvely representing a decision problem with a multilinear extension, which has "low" degree n and, having exponentially many terms, can take exponential time to compute, both of which are too large for our fine-grained analysis. A key technique we use to bypass the naïve construction, then, is to look at the structure of specific problems from fine-grained complexity to tailor *very* low-degree efficiently-computable polynomials to them. The matching upper and lower bounds are precisely what allows us to capture the complexity of our problems in the fine-grained setting and open the door for applications.

Extensions. We extend our results to a generalization of OV, called k-OV whose hardness can also be based on the SETH. To this end, we define a corresponding polynomial $\mathcal{F}$OV^k which is computable in $\widetilde{O}(n^k)$ time in the worst-case. Using the same ideas as in the above worst-case to average-case reductions we show:

- If k-OV requires $n^{k-o(1)}$ time to decide in the worst-case, then $\mathcal{F}$OV^k requires $n^{k-o(1)}$ time to evaluate with probability 3/4 on uniformly chosen inputs.

[1] Technically this requires *moderate-dimension* OV, for which our results still apply to, and we need to consider the generalization $\mathcal{F}$OV^k we introduce to account for *all* first-order graph properties.

We note that this yields a tight average-case time hierarchy: an average-case problem computable in time n^k but not much faster for every integer k (these can be extended to rational numbers through standard padding techniques). Unconditional average-case time hierarchies are known – e.g. [GGH94] – but these are based on canonical functions that have not found further use than as a proof of this concept, while our functions are closely related to well-studied problems and have considerable algebraic structure.

Building on [CPS99], we use local list decoding to show that our families of polynomials remain hard even when asking for their successful evaluation on just a $1/\text{polylog}(n)$ fraction of inputs. We extend this to attain a smooth trade-off between the running time and the upper bound on the probability of success of any average-case solver. We additionally show that $\mathcal{F}\mathsf{OV}$ remains hard to evaluate even over very small field sizes by applying an isolation lemma to OV, which may be of independent interest itself.

Applications. Leveraging the structure of our problems, and using ideas from [Wil16], we construct a *Proof of Work* scheme based on their average-case hardness. In subsequent work, [BRSV17] extends our Proofs of Work framework to introduce Proofs of *Useful* Work. That is, Proofs of Work whose completion is useful for solving practical problems. We pose the application of creating fine-grained cryptography as an open problem, and outline a structural barrier to achieving fine-grained one-way functions from our results.

Finally, we note that these reductions set up a win-win scenario: either the worst-case conjectures are true and we have average-case fine-grained hard problems *or* we can show them easy and thus break our worst-case conjectures, allowing a breakthrough in fine-grained complexity theory. We explore this notion further by considering the ideas introduced in [Nao03] of falsifiable assumptions to show that our results make the OV, $\mathsf{3SUM}$, and APSP conjectures falsifiable in a practical sense. Specifically, in Section 6.2 we show that *empirically* evaluating our polynomial for OV faster than we conjecture possible would give strong heuristic evidence that SAT has faster worst-case algorithms – i.e. that the SETH is false. In this sense, we discuss how our results allow for the *heuristic falsifiability* of conjectures.

1.2 Related Work

Recently, and independently of our work, a sequence of papers [BK16, GR17, Wil16] has also observed that a number of problems from the fine-grained world can be captured by the evaluation of efficiently computable low-degree polynomials. The focus of these papers has been on using the algebraic structure and low-degree of the polynomials to *delegate computation* of fine-grained problems in quickly verifiable ways. The papers were not, however, concerned with the hardness aspects of complexity and made no average-case claims or guarantees.

A key discovery, then, achieved here and independently in [BK16, GR17, Wil16], is the utility of looking at the structure of specific computational problems to tailor *very* low-degree polynomials that are efficiently computable to them. From the algorithmic perspective, [BK16, GR17, Wil16] find utility for delegation of computation, while, from the hardness perspective, we connect it to average-case complexity and applications thereof.

This gives a very rich framework that our results are applicable to, as our worst-case to average-case reductions can be adapted in a straightforward way to work for any problem appropriately expressible as a low-degree polynomial. Many other low-degree polynomials for interesting practical problems are found in [BK16, GR17, Wil16], with [Wil16] independently discovering our $\mathcal{F}\mathsf{OV}$ polynomial and [BK16] independently finding a polynomial similar to our $\mathcal{F}\mathsf{3SUM}$. The work of [GR17], in fact, identifies a natural class of "locally characterizable sets" that contains problems

admitting low-degree polynomials akin to the ones considered here. Many of the above polynomials can fit into the framework of applications of average-case fine-grained hardness, such as the Proofs of Work we introduce.

In subsequent work, [BRSV17] extends our Proofs of Work framework to obtain Proofs of *Useful* Work, which can be viewed as combining the delegation of computation from [BK16, GR17, Wil16] with the hardness results we achieve here. This allows for the ability to delegate arbitrary instances of practical problems yet guarantee that the response time will require some (tunable) fixed polynomial amount of time (even for easy instances). That is, Proofs of Work whose completion is useful for solving practical problems. Further shown in [BRSV17] is that the evaluation of our polynomials (and thus the Proofs of Work) are *non-amortizable* over multiple instances, even on average.

Lastly, [GH16] recently shows that fine-grained problems related to DNA sequencing are actually *easy* when given a batch of correlated instances. While these correlated instances are not typically what we consider in average-case complexity, they are distributional notions of inputs on fine-grained problems and so seems to be the closest existing work to average-case fine-grained complexity. The techniques used, however, are very different and focused on attaining *easiness* for specific problems with respect to specific distributions, whereas we focus on attaining hardness and applications of hardness and do so within the emerging low-degree polynomial framework. However, [GH16] can also be used to make claims similar to our notion of *heuristic falsifiability*, suggesting that whenever the intersection of average-case complexity and fine-grained complexity is considered it may immediately bear interesting fruit for this notion. We discuss this further in Section 6.2.

1.3 Organization

For reference, our paper's results are discussed as follows:

- Worst-case–to–average-case fine-grained reductions to the evaluation of our families of polynomials $\mathcal{F}\mathsf{OV}$, $\mathcal{F}\mathsf{3SUM}$, $\mathcal{F}\mathsf{ZWT}$, and $\mathcal{F}\mathsf{TC}$ in Section 3.1, Appendix C, Section 3.2, and Section 3.3, respectively.
- An infinite average-case time hierarchy arising from the assumption of SETH and two other hierarchies from the assumption of the k-SUM conjecture as discussed in Section 4.
- Definitions of fine-grained cryptography and one-way functions, a barrier to achieving them from our results, and a possible way to bypass this barrier all in Section 5.
- Constructions of Proofs of Work guaranteed by worst-case hardness assumptions in Section 6.1.
- Applications of our average-case results and Proofs of Work to the heuristic falsifiability of worst-case assumptions, including SETH, in Section 6.2.
- Amplifying hardness of our average-case problems and smooth trade-offs between hardness and running time in Appendix D.
- An isolation lemma for the Orthogonal Vectors problem that lets us perform our reductions to polynomials over much smaller fields in Appendix E.
- Open problems in average-case fine-grained hardness, fine-grained cryptography, and heuristically falsifying assumptions in practice in Section 7.

2 Worst-Case Conjectures

We consider a few well-studied problems in fine-grained complexity and conjectures about their worst-case hardness that we use to support our average-case hardness conjectures. For a more comprehensive survey of fine-grained complexity, connections between problems, and formal definitions of concepts like fine-grained reductions, see [Wil15].

(All our discussion will be in the Word RAM model of computation with $O(\log(n))$-bit words. When we speak of randomized algorithms in a worst-case setting, we mean algorithms that, for every input, output the correct answer with probability at least 2/3. And unless specified otherwise, all algorithms and conjectures about algorithms are randomized throughout the paper.)

2.1 Main Islands of Fine-Grained Complexity

First we recall the problems of OV, 3SUM, and APSP defined in Section 1.1. These problems currently remain the three key problems of fine-grained complexity as they partition what is known in the field. That is, there are no known reductions between them, but they reduce to many other problems and, thus, give us the basis for what we generally call hardness within P [Wil15]. This foundation is more formally given, after extensive attempts to find improved algorithms for them, through the following popular hardness conjectures:

- OV Conjecture: For any $d = \omega(\log n)$, any algorithm for OV_d requires $n^{2-o(1)}$ time.
- 3SUM Conjecture: Any algorithm for 3SUM requires $n^{2-o(1)}$ time.
- APSP Conjecture: Any algorithm for APSP requires $n^{3-o(1)}$ time.

These conjectures are not only important because they help stratify P, but the truth or falsity of each of them has many ramifications to practical problems.

It has been shown that if the the OV conjecture is true, then many string processing problems, hugely relevant to DNA sequencing and data comparison, also have hardness bounds (typically sub-quadratic) [AWW14, ABW15b, BI14, BK15]. On the other hand, if a sub-quadratic algorithm for OV is found, Williams [Wil05] gave a reduction to show we would achieve improved algorithms for SAT; more specifically, the well-known Strong Exponential Time Hypothesis (SETH) would break. Thus, as stated in Section 1.1, SETH implies the OV conjecture.

(We note that the results in this paper still go through under a slightly weaker variant of the OV conjecture: for all $\varepsilon > 0$, there is no $O(n^{2-\varepsilon}\mathsf{poly}(d))$ algorithm for OV_d. This problem, "Moderate Dimension" OV, was shown to be hard for the class of all first-order graph problems in [GI16].)

Similarly, if the 3SUM conjecture is true, problems in computational geometry [GO95] and exact weighted subgraph problems [AL13] are hard. Further, [AL13] shows that if the 3SUM conjecture is false, we get improved algorithms for many of the same graph problems.

Finally, the APSP conjecture's truth would give lower bounds for many problems in dense graphs [WW10] and for dynamic problems [RZ04]. [WW10] also shows that the conjecture being false gives better algorithms for the dense graph problems.

We note that while it is common to make these conjectures only for deterministic algorithms, we see these as structural beliefs about the problems – that brute force is essentially necessary as there is no structure to algorithmically exploit – and so there is no reason to believe that allowing randomness will allow a significant speed-up. Indeed, these conjecture are often made against randomized expected running time machines as in [Wil15] and randomized one-sided error versions of SETH have been made in [DHW10] and [CIKP03]. Further, [CFK+15] conjectures and argues for a SETH under randomized two-sided error machines (as we apply to all of our conjectures for the use of worst-case to average-case reductions).

Besides these three main islands of fine-grained complexity theory, a fourth seems to be emerging based on the k-CLIQUE problem. With the current best algorithm solving the problem in $n^{\omega k/3}$ time [NP85], where ω is the matrix multiplication constant, there has been recent work showing that conjecturing this to be optimal leads to interesting hardness results for other important problems such as parsing languages and RNA folding [ABW15a, BGL16, BDT16, BT16]. To explore delegating computation, [BK16, GR17, Wil16] all introduce different families of polynomials to express the k-CLIQUE problem, yet none yield analysis akin to those above. The polynomials either yield average-case hardness via our techniques but cannot be computed efficiently enough to give matching upper bounds [GR17, Wil16], or they have too large of a degree for our worst-case to average-case reductions [BK16]. We leave the open problem of finding a family of polynomials to represent k-CLIQUE that both are computable in time $n^{\omega k/3}$ and have degree $n^{o(1)}$.

2.2 Auxiliary Problems

For these practical connections, any reduction to or from the main island problems has interesting consequences (see [Wil15] for a more comprehensive treatment). In our results we achieve reductions *from* these problems and, to help facilitate that, we recall two more problems.

- *Zero-Weight Triangle:* Given an edge-weighted graph on n vertices, the ZWT problem is to decide whether there exists a triangle with edge weights w_1, w_2, w_3 such that $w_1 + w_2 = -w_3$, where edge weights are in $\{-n^c, \ldots, n^c\}$ for some sufficiently large c.
- *Triangle-Collection:* Given an graph on n vertices and a partition C of the vertices into colors, the Triangle-Collection problem is to decide whether for each triple of three colors $a, b, c \in C$, there exists vertices x, y, z in the graph that form a triangle and $x \in a$, $y \in b$, and $z \in c$. That is, each triplet of colors is 'collected' by some triangle.

We will follow the approach in [CGI+16] of, at times, using ZWT as a proxy for both 3SUM and APSP. That is, both reduce in a fine-grained way to ZWT: APSP reduces to (Negative Weight Triangle [WW10] and then to) ZWT [VW09], and 3SUM has a randomized (which suits our purposes) reduction to ZWT in [VW09, P̆10]. Thus reducing *from* ZWT reduces from both 3SUM and APSP simultaneously. It then follows that if *either* the 3SUM or APSP conjecture are true, then ZWT requires $n^{3-o(1)}$ time.

Similarly, the Triangle-Collection problem is introduced in [AWY15] as a way to base hardness on the believable conjecture that *at least one of* the SETH, 3SUM, or APSP conjectures are true. To do this, they give fine-grained reductions from *all three* of k-SAT, 3SUM, and APSP so that if any of their conjectures are true, then Triangle-Collection requires $n^{3-o(1)}$ time.

In general, it is better to reduce from problems furthest down a chain of reductions, as assuming those problems to be hard will then be the weakest assumption required - e.g. assuming Triangle-Collection requires $n^{3-o(1)}$ time is a *weaker* assumption than assuming that at least one of k-SAT, 3SUM, or APSP are hard. It is an interesting direction to base average-case fine-grained hardness on increasingly weaker assumptions.

For this reason, it would be desirable to reduce from some very practical DNA sequencing problems (e.g. EDIT-DISTANCE and LCS) that are reduced to *from* OV (and thus k-SAT). Further, there is mounting evidence that, regardless of the status of k-SAT's complexity, these DNA sequencing problems are in fact *very* likely to be hard [GI16, AHWW15]. We remark, however, that there is a barrier to representing these problems with low-degree polynomials [Abb17]. Namely, representing them with low-degree polynomials would allow for small speedups – i.e. by using the polynomial method [CW16] – but such speedups (of just shaving some logarithmic factors off of the runtime) have been show to imply new breakthroughs in circuit lower bounds [AHWW15].

3 Average-Case Fine-Grained Hardness

We now define the notion of average-case complexity that we shall use and describe the technique we use for our worst-case to average-case reductions. Then, we describe the problems we conjecture to be hard on average and show reductions from the worst-case problems described in Section 2 in support of these conjectures.

Definition 1. *A family of functions $\mathcal{F} = \{f_n\}$ is* computable in time t on average *if there is an algorithm that runs in $t(n)$ time on the domain of f_n and, for all large enough n, computes f_n correctly with probability at least $3/4$ over the uniform distribution of inputs in its domain.*

For broader definitions that are more useful when one is concerned with whole classes of problems rather than a handful of specific ones, and for extensive discussions of the merits of the same, we refer the reader to Bogdanov and Trevisan's survey [BT06].

To achieve average-case hardness for our fine-grained problems, our main technique will be to "express" these problems as low-degree polynomials and then use the random self-reducibility of evaluating these polynomials to attain average-case hard problems.

We now recall the classic random self-reducibility of evaluating low-degree polynomials, first used to show the average-case hardness of computing the Permanent when assuming its worst-case hardness [Lip91, FF90]. We can more generally view this as the local correctability of Reed-Muller codes first shown by Gemmell and Sudan [GS92] and get better error rates using techniques from this perspective. We repeat the proof in Appendix A in order to accurately assess the running time of the algorithm involved.

Lemma 1. *Consider positive integers N, D, and p, and an $\varepsilon \in (0, 1/3)$ such that $D > 9$, p is prime and $p > 12D$. Suppose that for some polynomial $f : \mathbb{F}_p^N \to \mathbb{F}_p$ of degree D, there is an algorithm A running in time t such that A is an average-case solver. That is,*

$$\Pr_{x \leftarrow \mathbb{F}_p^N} [A(x) = f(x)] \geq 1 - \varepsilon$$

Then there is a randomized algorithm B that runs in time $O(ND^2 \log^2 p + D^3 + tD)$ such that B is a probabilistic worst-case solver. That is, for any $x \in \mathbb{F}_p^N$:

$$\Pr [B(x) = f(x)] \geq \frac{2}{3}$$

Remark 1. The range of ε being $(0, 1/3)$ is arbitrary to some extent. It could be any constant smaller than $1/2$ at the cost of p having to be slightly larger.

Remark 2. An important thing to note here is how B's runtime depends on f's degree D. Assuming tD is the high-order term in the runtime, B runs in time $O(tD)$. So if we want our reductions to have low overhead, we will need D to be rather small. For our fine-grained purposes, we need to be careful in what we consider "low" and we will see that we always have D polylogarithmic in N.

We now introduce three families of polynomials that we conjecture average-case hard to evaluate and then give evidence for this by reducing to them from the worst-case problems OV, ZWT, and Triangle-Collection, respectively, and then applying the random self-reducibility of low-degree polynomials as just described. A fourth family of polynomials arising from 3SUM can be seen in Appendix C. The landscape of these reductions is seen in Figure 1.

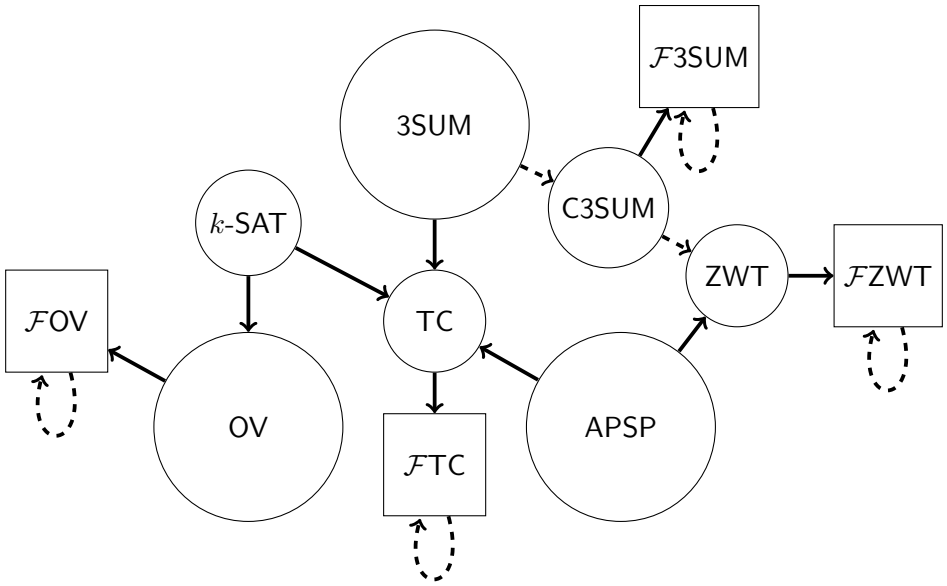

Figure 1: Arrows represent (fine-grained) reductions and dashed means they're randomized. Thus a dashed self-loop is a worst-case to average-case self-reduction. Our work introduces $\mathcal{F}$OV, $\mathcal{F}$3SUM, $\mathcal{F}$ZWT, and $\mathcal{F}$TC and the reductions involving them. See Appendix C for C3SUM and $\mathcal{F}$3SUM.

3.1 Orthogonal Vectors

For any n, let $p(n)$ be the smallest prime number larger than n^2, and $d(n) = \lceil \log^2 n \rceil$ (for brevity, we shall write just p and d). We define polynomials $f\mathsf{OV}_n : \mathbb{F}_p^{2nd} \to \mathbb{F}_p$ over $2nd$ variables. We view these variables as representing the input to OV – we separate the variables into two matrices $U, V \in \mathbb{F}_p^{n \times d}$. The polynomial $f\mathsf{OV}_n$ is then defined as follows:

$$f\mathsf{OV}_n(U,V) = \sum_{i,j \in [n]} \prod_{\ell \in [d]} (1 - u_{i\ell} v_{j\ell})$$

A similar polynomial was used independently by Williams [Wil16] to construct coMA proof systems for OV with efficient verifiers. Given an OV instance $(U,V) \in \{0,1\}^{2nd}$, $f\mathsf{OV}_n(U,V)$ counts the number of pairs of orthogonal vectors in it – for each pair $i,j \in [n]$, the corresponding summand is 1 if $\langle u_i, v_j \rangle = 0$, and 0 otherwise (there is no modular wrap-around of the sum as $p > n^2$). Also, $f\mathsf{OV}_n$ has degree at most $2d$, which is rather low.

Define the family of polynomials $\mathcal{F}\mathsf{OV} = \{f\mathsf{OV}_n\}$. We show a worst-case to average-case reduction from OV to $\mathcal{F}$OV that, given an algorithm that computes $f\mathsf{OV}_n$ well on average, decides OV on instances of length n without much overhead. This is stated as the following theorem.

Theorem 1. *If $\mathcal{F}$OV can be computed in $O(n^{1+\alpha})$ time on average for some $\alpha > 0$, then OV can be decided in $\widetilde{O}(n^{1+\alpha})$ time in the worst case.*

Proof. Suppose there were an algorithm A that ran in $O(n^{1+\alpha})$ time and computed $f\mathsf{OV}_n$ correctly on more than a 3/4 fraction of inputs for all large enough n.

In order to be able to use such an average-case algorithm, however, one has to be able to write down inputs to run it on. These inputs to $f\mathsf{OV}_n$ are in $\mathbb{F}_p^{2nd}$, and so to work with them it is necessary to know $p = p(n)$, the smallest prime number larger than n^2. Further, p would have to

be computable from n rather efficiently for a reduction that uses A to be efficient. As the following lemma states, this turns out to be possible to do.

Lemma 2 (Implied by [LO87])**.** *The smallest prime number greater than m can be computed deterministically in $\widetilde{O}(m^{1/2+\alpha})$ time for any $\alpha > 0$.*

We will then use A and this lemma to decide OV as follows. Given an input $(U, V) \in \{0,1\}^{2nd}$, first compute $p = p(n)$ – this can be done in $\widetilde{O}(n^{1+\alpha})$ time by Lemma 2. Once p is known, A can be used along with Lemma 1 to compute $f\mathsf{OV}_n(U, V)$ in $O(n(2d)^2 \log^2 p + (2d)^3 + 2dn^{1+\alpha}) = \widetilde{O}(n^{1+\alpha})$ time, and this immediately indicates membership in OV as observed above. □

Corollary 1. *If OV requires $n^{2-o(1)}$ time to decide, $\mathcal{F}\mathsf{OV}$ requires $n^{2-o(1)}$ time to compute on average.*

Note that our result is then *tight* under the OV conjecture in the sense that our polynomial is computable in $\widetilde{O}(n^2)$ time, but in no less (even on average) assuming sub-quadratic hardness of OV. That is, we demonstrate a problem that is quadratic-computable but sub-quadratic-hard on average. It should also be noted that our results can adapted the Moderate Dimension OV problem (as mentioned in Section 2) and thus an appropriately parametrized variant of $\mathcal{F}\mathsf{OV}$ is average-case hard for the class of all first-order graph problems as defined in [GI16].

3.2 3SUM and All-Pairs Shortest Path

Recall from Section 2 that both $\mathsf{3SUM}$ and APSP have fine-grained reductions to ZWT, and so we restrict our attention to ZWT. We now show a family of polynomials that can count Zero Weight Triangles.

For any n, let $p(n)$ denote the smallest prime number larger than n^3 and let $d = \lceil \log(2(2n^c + 1)) \rceil + 3$ (c being the constant from the definition of ZWT). We define the polynomial $f\mathsf{ZWT}_n : \mathbb{F}_p^{n^2 d} \to \mathbb{F}_p$ as taking in a set E of $n^2 d$ variables where we split them into n^2 sets, w_{ij}, of d variables each for all $i, j \in [n]$:

$$f\mathsf{ZWT}_n(E) = \sum_{i,j,k \in [n]} \prod_{\ell \in [d]} \left(1 - \left(s_\ell\left(w_{ij}, w_{jk}\right) - s_\ell\left(\overline{w}_{ik}, 0 \ldots 01\right)\right)^2\right)$$

where $s_\ell : \mathbb{F}_p^{2d} \to \mathbb{F}_p$ is the polynomial such that if $x, y \in \{0,1\}^d$, then $s_\ell(x, y)$ equals the ℓ^{th} bit of $(x + y)$ as long as x and y represent numbers in $[-n^c, n^c]$. Such polynomials exist, have degree at most $2d$, and are computable in $O(d \log^2 p)$ time – see Appendix B. Further, $\overline{w}_{ik}$ represents the set of linear polynomials that toggle all the bits in a boolean valued w_{ij}; so $s_\ell\left(\overline{w}_{ik}, 0 \ldots 01\right)$ effectively takes the one's complement of w_{ij} and then adds 1, which is exactly the two's complement of w_{ij}.

Now, considering a graph on n vertices with edges weighted from $[-n^c, \ldots, n^c]$. We use this polynomial to count zero weight triangles in it: For an edge-weight between nodes i and j we decompose the value to its bit representation in two's complement notation and now have d boolean inputs for w_{ij}. If an edge does not exist between an i and j, we similarly put the bit decomposition of the value $2n^c + 1$ into w_{ij} (note that $i = j$ is possible and we consider there to *not* be an edge for this). Conceptually, we now have weights w_{ij} corresponding to a complete graph on n vertices with the the non-edges added at weight $2n^c + 1$. Note that each triangle in it is zero weight if and only if it was a zero weight triangle in the original graph. Thus, collecting these all together we have boolean input $E \in \{0,1\}^{n^2 d}$. This reduction certainly takes sub-cubic time.

Then, given the binary representation of a ZWT instance, the ℓ^{th} term in the product above checks whether the ℓ^{th} bit of the sum of w_{ij} and w_{jk} equals that of the negation of w_{ik}. If all d bits

are equal, then, and only then, the summand is 1, otherwise it is 0. So the sum counts the number of triples of distinct (i, j), (j, k), and (i, k) such that $w_{ij} + w_{jk} = -w_{ik}$. Also, the degree of $f\mathsf{ZWT}_n$ is at most $4d^3 = O(\log^3 n)$.

Define the family of polynomials $\mathcal{F}\mathsf{ZWT} = \{f\mathsf{ZWT}_n\}$. The following theorem can be proved identically to Theorem 1.

Theorem 2. *If $\mathcal{F}$ZWT can be computed in $O(n^{1.5+\alpha})$ time on average for some $\alpha > 0$, then* ZWT *can be decided in $\widetilde{O}(n^{1.5+\alpha})$ time in the worst case.*

Corollary 2. *If* ZWT *requires $n^{3-o(1)}$ time to decide, $\mathcal{F}$ZWT requires $n^{3-o(1)}$ time to compute on average.*

Thus, assuming the ZWT conjecture, using the fact that $f\mathsf{ZWT}_n$ has n^3 terms and each s_ℓ is computable in $O(d)$ time, we again achieve tightness where $\mathcal{F}$ZWT is cubic-computable but sub-cubic-hard. It is also worth noting that the following corollary frames our result in the more familiar problems of 3SUM and APSP.

Corollary 3. *If* either 3SUM *requires $n^{2-o(1)}$ time or* APSP *requires $n^{3-o(1)}$ time, then $\mathcal{F}$*ZWT *takes $n^{3-o(1)}$ time to compute on average.*

3.3 SETH, 3SUM, and All-Pairs Shortest Path

We now give our most encompassing worst-case–to–average-case result. Recall from Section 2 that if *any* of k-SAT, 3SUM, or APSP are hard then the Triangle-Collection problem is also hard [AWY15], thus so would be any polynomial based on it. We can hence focus our attention on Triangle-Collection. More specifically, we will look at a restricted version of the problem called Triangle-Collection* shown to be equivalent to Triangle-Collection in [AWY15, Abb17], whose extra structure we will use to construct low-degree polynomials.

- *Triangle-Collection*:* Given an undirected tripartite node-colored graph G with n colors and $m = n\log^2 n + 2n\log^4 n$ nodes and with partitions A, B, C of the form:
  - A contains $n\log^2 n$ nodes $a_{\ell,i}$ where $i \in [n], \ell \in [\log^2 n]$ and $a_{\ell,i}$ is colored with color i.
  - B (respectively C) contains $n\log^4 n$ nodes $b_{\ell,i,x}$ (respectively $c_{\ell,i,x}$) where $i \in [n], \ell \in [\log^2 n], x \in [\log^2 n]$ and $b_{\ell,i,x}$ (respectively $c_{\ell,i,x}$) is colored with color i.
  - For each node $a_{\ell,i}$ and colors $j, k \in [n]$, there is exactly one edge from A to B of the form $(a_{\ell,i}, b_{\ell,j,x})$ and exactly one edge from A to C of the form $(a_{\ell,i}, c_{\ell,k,y})$, for some $x, y \in [\log^2 n]$.
  - A node $b_{\ell,j,x}$ can only be connected to nodes of the form $c_{\ell,k,y}$ in C. (There no edges across disparate ℓ's.)

  For all triples of distinct colors i, j, k, is there a triangle (u, v, w) in G where u has color i, v has color j, and w has color k?

We now give a polynomial whose evaluation would allow us to decide Triangle-Collection*. For any n, let $p(n)$ denote the smallest prime number larger than n^3. We define the polynomial $f\mathsf{TC}_n : \mathbb{F}_p^m \to \mathbb{F}_p$ as taking in a set E of $m = (n\log^2 n + 2n\log^4 n)^2$ variables (corresponding to entries in the adjacency matrix of an input graph to the above problem):

$$f\mathsf{TC}_n(E) = \sum_{1\le i<j<k\le n} \prod_{\substack{\ell,x,y\in[\log^2 n]\\ \pi\in S_{\{i,j,k\}}}} \left(1 - e_{a_{\ell,\pi(i)},b_{\ell,\pi(j),x}} e_{a_{\ell,\pi(i)},c_{\ell,\pi(k),y}} e_{b_{\ell,\pi(j),x},c_{\ell,\pi(k),y}}\right)$$

(Note that for a set X, S_X denotes the set of permutations on X.)

Consider a tripartite graph as defined above with adjacency matrix E. For each triple of colors, $(i,j,k) \in [n]^3$, if there is a corresponding triangle in the graph then it zeroes out that particular term, otherwise it will evaluate to one. Thus, $f\mathsf{TC}_n$ counts the number of colors not collected by a triangle – i.e. the number of violations to being a YES instance – and so, for boolean E, $f\mathsf{TC}_n(E) = 0$ if and only if E corresponds to a YES instance of Triangle-Collection*. Moreover, the degree of $f\mathsf{TC}_n$ is at most $18\log^6 n$.

Define the family of polynomials $\mathcal{F}\mathsf{TC} = \{f\mathsf{TC}_n\}$. The following theorem can be proved identically to Theorem 1.

Theorem 3. *If $\mathcal{F}\mathsf{TC}$ can be computed in $O(n^{1.5+\alpha})$ time on average for some $\alpha > 0$, then TC can be decided in $\widetilde{O}(n^{1.5+\alpha})$ time in the worst case.*

Corollary 4. *If TC requires $n^{3-o(1)}$ time to decide, $\mathcal{F}\mathsf{TC}$ requires $n^{3-o(1)}$ time to compute on average.*

Thus, $f\mathsf{TC}_n$ only having n^3 many summands with each being computable in polylog(n) time, it is easily seen that we again achieve tightness where $\mathcal{F}\mathsf{TC}$ is cubic-computable but sub-cubic-hard. More recognizably we attain the following.

Corollary 5. *If* either SETH *holds,* $\mathsf{3SUM}$ *takes $n^{2-o(1)}$ time,* or APSP *takes $n^{3-o(1)}$ time, then $\mathcal{F}\mathsf{TC}$ takes $n^{3-o(1)}$ time to compute on average.*

Note that this does not subsume the hardness of $\mathcal{F}\mathsf{ZWT}$ as, even if SETH fails and $\mathsf{3SUM}$ and APSP become easy, the ZWT problem may still be hard and yield hardness for $\mathcal{F}\mathsf{ZWT}$.

4 An Average-Case Time Hierarchy

In this section we present an infinite collection of generalizations of $\mathcal{F}\mathsf{OV}$ that we conjecture form an average-case time hierarchy. That is, for every rational number k the collection contains a function $\mathcal{F}\mathsf{OV}^k$ such that $\mathcal{F}\mathsf{OV}^k$ is computable in $\widetilde{O}(n^k)$ time, but (we conjecture) requires $n^{k-o(1)}$ time to compute even on average. This conjecture is supported by SETH. We describe these generalizations below and indicate how this follows from SETH for integer values of $k \ge 2$ and note that this can be extended to all rational numbers using standard padding techniques.

- *k-Orthogonal Vectors:* For an integer $k \ge 2$, the k-OV problem on vectors of dimension d is to determine, given k sets $(U_1, \ldots, U_k)$ of n vectors from $\{0,1\}^{d(n)}$ each, whether there exist $u_i \in U_i$ for each i such that over $\mathbb{Z}$,

$$\sum_{\ell\in[d(n)]} u_{1\ell}\cdots u_{k\ell} = 0$$

  (As with OV, if left unspecified, d is to be taken to be $\lceil\log^2 n\rceil$.)

Similar to how it implies the hardness of OV, (the randomized version of) SETH also implies that for any integer $k \geq 2$, any randomized algorithm for k-OV requires $n^{k-o(1)}$ time – the proof is a natural generalization of that for OV shown in [Wil15]. We next take the same approach we did for OV and define for any integer $k \geq 2$ a family of polynomials $\mathcal{F}\mathsf{OV}^k = \left\{f\mathsf{OV}_n^k\right\}$, where with p being the smallest prime number larger than n^k and $d = \lceil \log^2(n) \rceil$, $f\mathsf{OV}_n^k : \mathbb{F}_p^{knd} \to \mathbb{F}_p$ is defined as:

$$f\mathsf{OV}_n^k(U_1, \ldots, U_k) = \sum_{u_1 \in U_1, \ldots, u_k \in U_k} \prod_{\ell \in [d]} (1 - u_{1\ell} \cdots u_{k\ell})$$

Exactly as $f\mathsf{OV}_n$ does, when $f\mathsf{OV}_n^k$'s input is a k-OV instance from $\{0,1\}^{knd}$, $f\mathsf{OV}_n^k(U_1, \ldots, U_k)$ counts the number of sets of "orthogonal" vectors in it. Note that the degree of $f\mathsf{OV}_n^k$ is at most kd. And also that by simply evaluating each summand and adding them up, the polynomial can be evaluated in $\widetilde{O}(n^k)$ time. The following theorem can again be proven in a manner identical to Theorem 1.

Theorem 4. *For any integer $k \geq 2$, if $\mathcal{F}\mathsf{OV}^k$ can be computed in $O(n^{k/2+\alpha})$ time on average for some $\alpha > 0$, then k-OV can be decided in $\widetilde{O}(n^{k/2+\alpha})$ time in the worst case.*

Corollary 6. *Suppose for every $k \geq 2$, k-OV requires $n^{k-o(1)}$ time to decide. Then for every such k, $\mathcal{F}\mathsf{OV}^k$ requires $n^{k-o(1)}$ to compute on average but can be computed in the worst case in $\widetilde{O}(n^k)$ time.*

Thus, we can attain the hierarchy from an infinite number of conjectures, one for each k, but, as noted earlier, the entire hierarchy is also implied by the single assumption SETH. We note that for k-OV (and all other problems) we could naïvely express k-OV with a polynomial via a multilinear extension, yet this polynomial, as discussed in Section 1.2, would have exponentially many terms and degree n. Already the degree is too high for our purposes but not by much: we may not be too disappointed with $n^{(k-1)-o(1)}$ average-case hardness that degree n would still afford us. The main problem then is that the naïve polynomial may take exponential time to compute and so the upper bound is *very* far from the lower bound. The tightness of our hierarchy is a key feature then in capturing the hardness of our problems as well as for use in applications such as in Section 6.1 and in [BRSV17].

Remark 3. We can also attain two semi-tight hierarchies from generalizations of the 3SUM problem. That is, if we assume the k-SUM conjecture proposed in [AL13], we get hardness for two infinite hierarchies, with one based on generalizing $\mathcal{F}$ZWT and one from generalizing a polynomial introduced in Appendix C, $\mathcal{F}$3SUM (the proper generalizations can be based on problems found in [AL13]). These hierarchies, however, are loose, in that the k-SUM conjecture gives us $\left(n^{\lceil k/2 \rceil - o(1)}\right)$ hardness at the k^{th} level but, to our knowledge, our generalized polynomials are only $\widetilde{O}(n^{k-1})$ computable (as they have n^{k-1} many terms).

5 Towards Fine-Grained Cryptography

Average-case hardness has frequently found use in cryptography, where it is in fact a necessity – it is almost always required that a cryptographic object be hard to defeat on average for it to be useful. In this light, it is a natural question to ask whether the worst-case to average-case reductions presented here, along with the conjectured hardness of the worst-case problems, can be used to do cryptography.

Of course, since these problems are actually computable in polynomial time, one cannot expect to use them to construct standard cryptographic objects, which have to be secure against all polynomial time adversaries. We hence consider *fine-grained* cryptography, where we only ask for security against adversaries that run in some fixed polynomial time.

A major reason for interest in such notions is that a form of cryptography might be realizable even in Impagliazzo's Pessiland (or Heuristica or even Algorithmica) [Imp95]. That is, even if we live in a world where One-Way Functions or "coarse-grained" average-case hardness do not exist, it may still be possible to construct fine-grained cryptographic objects and salvage practical cryptography. Further, even if we *do* have standard cryptography, it may be the case that fine-grained cryptography can use weaker assumptions as it only needs to be secure against moderately powerful adversaries. Though not done so very extensively and not done from the sort fine-grained complexity in [Wil15], such notions have indeed been considered several times before – see for instance [Mer78, Hås87, Mau92, DVV16].

In this section we define a notion of fine-grained cryptography in Section 5.1, show a structural barrier to achieving it in Section 5.2, and outline as an open problem a possible approach to circumventing this barrier in Section 5.3. While this outline has not yet yielded fine-grained cryptography, we use its techniques in Section 6.1 to obtain Proofs of Work. Our framework has since been extended to achieve Proofs of Useful Work [BRSV17], which we also mention in Section 6.1

5.1 Fine-Grained One-Way Functions

To illustrate the kinds of objects, approaches, and difficulties fine-grained cryptography might involve, we consider the case of One-Way Functions (OWFs). A fine-grained OWF would capture the same concept as a standard OWF – easy to compute yet hard to invert – but with a more fine-grained interpretation of "easy" and "hard".

Definition 2. *We say a function* $f : \{0,1\}^* \to \{0,1\}^*$ *is* (t,ϵ)-one-way *if it can be computed in* $O(t(n)^{1-\delta})$ *time for some* $\delta > 0$*, but for any* $\delta' > 0$*, any* $O(t(n)^{1-\delta'})$*-time algorithm* A*, and all sufficiently large* n*,*

$$\Pr_{x \leftarrow \{0,1\}^n}\left[A(f(x)) \in f^{-1}(f(x))\right] \leq \epsilon(n,\delta')$$

One approach to constructing such OWFs (that has perhaps been part of folklore for decades) is as follows. Take one of our hard-on-average to evaluate polynomials – e.g. $f\mathsf{OV}_n$ – and suppose there was an input-output sampling algorithm $S \equiv (S_1, S_2)$ that runs in sub-quadratic time in n such that, on uniform input r, $S_1(r)$ is distributed uniformly over the appropriate domain, and $S_2(r) = f\mathsf{OV}_n(S_1(r))$. By our results it can be seen that S_1 is $(n^2, 3/4)$-one-way if we assume the OV conjecture (strengthened to assume hardness for all sufficiently large input sizes).

5.2 Barriers and NSETH

However, as stated, there turn out to be certain barriers to this approach. For instance, if $S(r) = (x,y)$, then r would be a certificate that $f\mathsf{OV}_n(x) = y$ that can be verified in sub-quadratic time. In particular, if $x \in \{0,1\}^{2nd}$ is a NO instance of OV, then r is a certificate for this that is verifiable in deterministic sub-quadratic time – i.e. that $f\mathsf{OV}_n(x) = 0$.

Further, tracing this back through the reduction of k-SAT to OV (see [Wil15]), this gives us a certificate for NO instances of k-SAT (for any k) that are verifiable in $O(2^{n(1-\epsilon)})$ time for some $\epsilon > 0$ – i.e. short and quickly verifiable certificates for $\mathsf{CNF\text{-}UNSAT}$ instances. Interestingly, the impossibility of this was recently conjectured and formalized as NSETH (the Non-deterministic

Strong Exponential Time Hypothesis) in [CGI+16] along with the establishment of its barrier status: its falsification would yield breakthroughs in both circuit complexity and proof complexity.

An alternative view of matters would be that this presents another approach to breaking NSETH. In fact, something weaker would suffice for this purpose – one only needs a sampler that runs in sub-quadratic time and samples $(x, f\mathsf{OV}_n^k(x))$ for some k such that the distribution of x has $\{0,1\}^{knd}$ in its support.

Note that while a sampler based on OV would, because of its relation to k-SAT, break NSETH, a sampler based on ZWT would only yield quick deterministic certification for APSP instances (the reduction from 3SUM is randomized). So we are left with much weaker "barriers" for $\mathcal{F}$3SUM and $\mathcal{F}$ZWT samplers. Indeed, [CGI+16]'s introduction of NSETH was in a direction opposite to ours: while we explore NSETH to argue that OV is unlikely to have small co-nondeterministic complexity because of its relationship to SAT, they introduce it to argue that 3SUM and APSP are unlikely to have a relationship to SAT (as OV does) by showing them to actually have small co-nondeterministic complexities. Thus they explicitly break both "barriers" for a $\mathcal{F}$3SUM or $\mathcal{F}$ZWT sampler!

Still, $\mathcal{F}$OV is much simpler algebraically and so seems to hold more hope for constructing a fine-grained OWF in this manner. We now discuss ways in which we may still salvage a sampler for $\mathcal{F}$OV.

5.3 A Way Around

There are visible ways to skirt this NSETH barrier: Suppose that the sampler S was not perfect – that with some small probability over r it outputs (x, y) such that $f\mathsf{OV}_n(x) \neq y$. Immediately, NSETH no longer applies, as now an r such that $S(r) = (x, y)$ may no longer be a sound certificate that $f\mathsf{OV}_n(x) = y$. And, as explained below, such a sampler still gives us a fine-grained *distributional* OWF based on the hardness of OV as long as the probability that it is wrong is small enough.

Informally, a distributional OWF is a function f such that $f(x)$ is easy to compute, but for most y's it is hard to sample a (close to) uniformly random x such that $f(x) = y$. A distributional OWF might not be a OWF itself, but it is known how to derive a OWF from any distributional OWF [IL89]. And further, this transformation works with quasi-linear overhead using constructions of hash functions from [IKOS08].

We claim that S_1 is now a fine-grained distributional OWF. Intuitively, if it were not, then we would, for many x, be able to sub-quadratically sample r almost uniformly from those obeying $S_1(r) = x$. But, since the probability over r is low that S errs, the r we sampled is likely a non-erring one. That is, it is likely that we would have sub-quadratically obtained an r that gives us $S_2(r) = f\mathsf{OV}_n(x)$ and thus we likely can sub-quadratically compute $f\mathsf{OV}_n(x)$. So if $f\mathsf{OV}_n$ is actually hard to compute on average, then such a distributional inverter cannot exist.

While, as mentioned earlier, such an erring sampler would no longer give efficiently verifiable certificates for NO instances of OV, it turns out that it would still yield a "barrier" of a coAM protocol for OV with sub-quadratic verification.

First we note that we get an AM protocol with a sub-quadratic time verifier that takes input (x, y) and proves that $f\mathsf{OV}_n(x) = y$, and is complete and sound *for most values* of x. Since the sampler is wrong with only with a very small probability over r, most values of x's have the property that most values of r satisfying $S_1(r) = x$ also satisfy $S_2(r) = f\mathsf{OV}_n(x)$. In the protocol with input (x, y), the verifier simply asks the prover to prove that for most r's such that $S_1(r) = x$, it is also the case that $S_2(r) = y$. If S_1 is indeed distributed uniformly, then this comes down to proving a lower bound on the number of r's such that $S_1(r) = x$ and $S_2(r) = y$, and this can be done in AM using the protocol from [GS86], in a single round (verifier sends a message and prover responds)

and with the verifier running in sub-quadratic time. With a little more analysis and appropriate setting of parameters, this protocol can be shown to work even if S_1 is only close to uniform.

Further, from this protocol one can get a protocol that works *for all values* of x by following the approach in the proof of Lemma 1 – given an input (x, y), the verifier runs the random self-reduction for x and, for each evaluation query to $f\mathsf{OV}_n$ that comes up in its course, it asks the prover for the answer along with an AM proof of its correctness. Thus, done all at once, this gives a single-round coAM protocol for OV with a sub-quadratic time verifer (this is when $y = 0$). This in turns leads to a single-round coAM protocol for k-SAT with a verifier that runs in $\widetilde{O}(2^{n(1-\epsilon)})$ time for some $\epsilon > 0$.

The impossibility of the existence of such a protocol could be conjectured as a sort of $\mathsf{AM}[2]\mathsf{SETH}$. Williams [Wil16] gives a non-interactive MA protocol that comes close to breaking this conjecture, but standard approaches for converting MA protocols to AM protocols [Bab85] seem to incur a prohibitively large overhead in our fine-grained setting.

We next detail the MA protocol from [Wil16] to pose an open problem of efficiently converting this to an AM protocol, thus breaking the barrier to an erring sampler (and thus a fine-grained OWF) with the further hope such a sampler could be "reverse-engineered" from an AM protocol. In Section 6.1, show how this MA protocol is already useful to our framework by constructing Proofs of Work.

5.4 An MA Protocol

We briefly describe the MA protocol for $f\mathsf{OV}_n$ achieved in [Wil16], where a more general protocol and framework can be found. Recall that the objective of the prover is to convince the verifier that, for given (x, y), $f\mathsf{OV}_n(x) = y$. We will expand x into $(U, V) \in \mathbb{F}_p^{n\times d} \times \mathbb{F}_p^{n\times d}$, which is the syntax we used when defining $f\mathsf{OV}_n$ as:

$$f\mathsf{OV}_n(U, V) = \sum_{i,j\in[n]} \prod_{\ell\in[d]} (1 - u_{i\ell}v_{j\ell})$$

For a fixed V, we define the supporting polynomial $f\mathsf{OV}_V : \mathbb{F}_p^d \to \mathbb{F}_p$ as:

$$f\mathsf{OV}_V(x_1, \ldots, x_d) = \sum_{j\in[n]} \prod_{\ell\in[d]} (1 - x_\ell v_{j\ell})$$

Now we may write $f\mathsf{OV}_n$ as:

$$f\mathsf{OV}_n(U, V) = \sum_{i\in[n]} f\mathsf{OV}_V(u_{i1}, \ldots, u_{id})$$

Next let $\phi_1, \ldots, \phi_d : \mathbb{F}_p \to \mathbb{F}_p$ be the polynomials of degree $(n-1)$ such that for $i \in [n]$, $\phi_\ell(i) = u_{i\ell}$ (treating i as an element of $\mathbb{F}_p$). Define the polynomial $R_{U,V}(x) = f\mathsf{OV}_V(\phi_1(x), \ldots, \phi_d(x))$. We can then write:

$$f\mathsf{OV}_n(U, V) = \sum_{i\in[n]} R_{U,V}(i)$$

The degree of $R_{U,V}$ is at most $(n-1)d$. If the verifier knew the coefficients of $R_{U,V}$, then it could evaluate it on the points $\{1, \ldots, n\}$ in time $\widetilde{O}(nd)$ (using fast techniques for batch evaluation on univariate polynomials [Fid72]) and thus compute $f\mathsf{OV}_n(U, V)$. This suggests the following protocol. The prover sends over the coefficients of a polynomial R^* that it claims to be $R_{U,V}$.

To verify this claim, the verifier checks whether $R^*(x) = R_{U,V}(x)$ for a random $x \in \mathbb{F}_p$. This can be done because even though the verifier does not know the coefficients of $R_{U,V}$, it can evaluate it at an input x in time $\widetilde{O}(nd)$ by first evaluating all the $\phi_\ell(x)$'s (these polynomials can be found using fast interpolation techniques for univariate polynomials [Hor72]), and then evaluating $f\mathsf{OV}_V$ using these values. By the Schwartz-Zippel lemma, since the field is considerably larger than the degree of R^* and $R_{U,V}$, if these are not the same polynomial, the verifier will catch this with high probability. If this check passes, the verifier uses R^* to compute $f\mathsf{OV}_n(U, V)$.

We pose the open problem of using the ideas of this protocol to construct an AM protocol that does the same. Further we pose the problem of creating an erring sampler from such a protocol, or more generally constructing a fine-grained OWF. We show in Section 6.1 how to use the existing MA scheme to create Proofs of Work.

6 Proofs of Work and the Falsifiability of Conjectures

In this section we present two other applications of our results. While we focus on $\mathcal{F}\mathsf{OV}$, these applications can similarly be made for $\mathcal{F}\mathsf{3SUM}$, $\mathcal{F}\mathsf{ZWT}$, and $\mathcal{F}\mathsf{TC}$ (and $\mathcal{F}\mathsf{OV}^k$ from Section 4).

6.1 Proofs of Work

A Proof of Work (PoW) scheme is a means for one party (that we call the *Prover*) to prove to another (called the *Challenger*) that it has expended a certain amount of computational resources. These were introduced by Dwork and Naor [DN92] as a means to deter spam mail and denial-of-service attacks, and have also found use in cryptocurrencies [Nak08]. Here we formulate a simplified definition of a PoW scheme to illustrate how our results could be used in this context. A PoW scheme consists of three algorithms:

- $\mathsf{Challenge}(1^n)$ takes a difficulty parameter n and produces a challenge c of this difficulty.
- $\mathsf{Solve}(c)$ solves the challenge c and produces a solution s.
- $\mathsf{Verify}(c, s)$ verifies that s is a valid solution to the challenge c.

The idea is that when the challenger wants the prover to work for a certain amount of time (and prove that it has done so), it runs $\mathsf{Challenge}$ with the appropriate difficulty parameter to generate a challenge c that it sends over. The prover is then required to produce a solution s to the challenge that the challenger verifies using Verify.

Intuitively, our average-case hardness results for $f\mathsf{OV}_n$ says that random inputs, thought of as challenges, must require a certain amount of work for a prover to evaluate $f\mathsf{OV}_n$ on. Then, the MA protocol in Section 5.4 gives a quick way for the challenger to verify a solution.

Definition 3. *The triple of algorithms* $(\mathsf{Challenge}, \mathsf{Solve}, \mathsf{Verify})$ *is a Proof of Work (PoW) scheme if the following properties are satisfied:*

- $\mathsf{Challenge}$ *and* Verify *are computable in time* $s(n)$.
- Solve *is computable in time* $t(n)$ *and* $\mathsf{Verify}(c, \mathsf{Solve}(c)) = 1$. *(*$t(n) > s(n)$*)*
- *There are constants* $\epsilon, \epsilon' \in [0, 1]$ *such that for any* A *and* n *satisfying:*

$$\Pr_{c \leftarrow \mathsf{Challenge}(1^n)} [\mathsf{Verify}(c, A(c)) = 1] \geq \epsilon$$

it is the case that A *takes time at least* $t'(n)$ *on a* ϵ' *fraction of challenges of difficulty* n. *(*$t'(n) > s(n)$. *Ideally,* $t'(n) \approx t(n)$.*)*

We want Challenge and Verify to be efficiently computable so that the challenger does not have to do too much work in this process. We want Solve to be more expensive but not by too much so that the prover can indeed produce a valid solution, albeit with more work than that done by the challenger. The lower bound on the difficulty of producing valid solutions ensures that the prover actually has to spend a certain amount of time on this task and that its production of a valid solution is proof that it has spent this time.

While above we use time as the computational resource of interest as we are working in the Word RAM model, these specifications can be in terms of other resources such as space, circuit size, etc. when working in other models. For different applications, there could also be other properties that might be desirable in such schemes such as resistance to parallelism, non-amortizability, etc., but we shall ignore these for now.

The MA protocol for $f\mathsf{OV}_n$ along with our average-case reductions can be used to construct a PoW scheme based on the hardness of OV as follows.

- $\mathsf{Challenge}(1^n)$ samples a random input $(U, V) \in \mathbb{F}_p^{2nd}$ to $f\mathsf{OV}_n$.
- $\mathsf{Solve}((U, V))$ outputs the coefficients of $R_{U,V}$ (as defined in Section 5.4).
- $\mathsf{Verify}((U, V), R^*)$ checks that $R^*(x) = R_{U,V}(x)$ for a random $x \in \mathbb{F}_p$.

By the arguments in Sections 3.1 and 5.4 regarding the verifier in the MA protocol, both Challenge and Verify above can be computed in $\widetilde{O}(n)$ time. Solve can be computed in $\widetilde{O}(n^2)$ time either by using the explicit expression for $R_{U,V}$, or by evaluating $R_{U,V}$ on sufficiently many points using the ϕ_ℓ's and $f\mathsf{OV}_V$ and interpolating to find its coefficients.

On the other hand, if any algorithm A produces R^* that passes Verify with probability, say, $5/6$ (over the uniform distribution of (U, V)), then by the soundness of Verify it follows that A is actually producing $R_{U,V}$ with almost the same probability. As noted earlier, given the coefficients of $R_{U,V}$, $f\mathsf{OV}_n(U, V)$ can be computed in $\widetilde{O}(nd)$ time. So A can be used to compute $f\mathsf{OV}_n$ correctly on average with an additive $\widetilde{O}(nd)$ overhead.

Corollary 1 now implies that if OV is hard for sub-quadratic algorithms, then A cannot be sub-quadratic *and* correct on more than a $3/4$ fraction of inputs. This leaves at least a $\frac{5}{6} - \frac{3}{4} = \frac{1}{12}$ fraction of inputs where A takes $\Omega(n^{2-o(1)})$ time. These numbers can be improved by repetition and by considering the better reductions from Appendix D.

Thus, based on the conjectured hardness of OV, the above is a PoW scheme where a party can prove to a challenger that it has run for a certain amount of time where the challenger only has to run for about a square-root of this amount of time.

This framework has since been used to create Proofs of Useful Work [BRSV17]. By allowing the ability to delegate arbitrary instances of OV (or other fine-grained problems) while still guaranteeing that the challenges will be hard on average (even for easy instances), [BRSV17] achieves Proofs of Work from our framework whose completion is useful for solving practical problems.

Remark 4. We note that OV is non-amortizable due to downward self-reducibility. For instance, let $\ell(n) = O(n^{1-\varepsilon})$ for some $\varepsilon > 0$. If there is a batch evaluation algorithm that can solve $\ell^2(n)$ instances of size $n/\ell(n)$ in time at most $O(n^{2-\delta})$ for some $\delta > 0$, then OV on an input (U, V) can be solved in sub-quadratic time by partitioning the U and V into $\ell(n)$ sets of size $n/\ell(n)$ each and running this algorithm on all pairings of these partitions together.

This has recently been extended [BRSV17] to show that $f\mathsf{OV}_n$ (as well as the above PoW) is non-amortizable in the average-case – i.e. if there is a sub-quadratic *batch* evaluation algorithm for uniformly random $f\mathsf{OV}_n$ instances, then OV can be solved in sub-quadratic time using the above approach along with ideas from the reduction in Section 3.1.

6.2 On the Heuristic Falsifiability of Conjectures

The above Proof of Work yields a win-win in the domain of algorithms and complexity. Namely, either we have a PoW *or* we have the existence of a *provably* sub-quadratic prover that can convince the challenger with sufficient probability which will in turn yield breakthroughs: a randomized sub-quadratic time algorithm for OV and thus a randomized $2^{n(1-\epsilon)}$ time algorithm for k-SAT for some $\epsilon > 0$. In particular, because the hardness of the PoW is over random instances, even a prover that can be *empirically* demonstrated to be sub-quadratic in practice will give *heuristic* evidence that the conjectured hardness of OV or k-SAT is false. In other words, if the above PoW is empirically (after working out the constants hidden by Big-Oh notation) insecure, then (randomized) SETH is heuristically falsified.

This notion of heuristic falsifiability sits directly at the marriage of average-case and fine-grained complexity that we accomplish: Without average-case results, a worst-case conjecture could not be broken without a full proof that an algorithm works on *all* inputs, and, without fine-grained results, a conjecture on large time complexity like SETH could not be feasibly be broken in practice. Thus, it is *precisely* average-case fine-grained hardness that allows us to discuss the heuristic falsifiability of conjectures. While theory and practice can often influence each other indirectly, this marks an interesting connection, akin to the hard-sciences, in which empirical evidence can give concrete and parameterizable theoretical evidence.

Remark 5. We note that while some may try to claim that SETH is already being falsified in practice - e.g. that we might seem to run in $2^{\sqrt{n}}$ time on practical inputs - there are two main points in which this is different than our heuristic falsifiability. One point is that, from our worst-case to average-case reductions, our notion would be heuristically breaking the *worst-case* version of SETH and achieve a complexity theoretic claim, as opposed to heuristically breaking an *average-case* notion of SETH on "nature's distribution" of "practical" inputs, which only says a heuristic claim on how we perform in practice. Secondly, claims of breaking even such an *average-case* notion of SETH in practice cannot be given too much confidence to since the input sizes must remain very small to be feasible and so not many "data points" can be gathered to see the true shape of the exponential curve. In contrast, our heuristic falsifiability reduces to a quadratic time problem, so that many more "data points" can be gathered for runtimes on much larger input sizes, giving us much more confidence as to if we heuristically break OV's conjecture and thus SETH.

To put this observation in more concrete terms, we consider [Nao03] in which Moni Naor introduced a stronger notion of falsifiable assumptions: Informally, an assumption A is efficiently falsifiable if there is an efficiently samplable distribution of challenges and an efficient verifier of solutions to these challenges such that, if and only if A is false (we consider the "only if" case), there is an efficient algorithm which causes the verifier to accept with high probability over the challenge distribution.

In our case, any algorithm that solves $\mathcal{F}$OV in sub-quadratic time falsifies the conjectured hardness of $\mathcal{F}$OV and thus OV and thus SETH. The sampler simply uniformly draws a set of inputs for $f\mathsf{OV}_n$, and the verifier simply evaluates $f\mathsf{OV}_n$ on the instances and compares with the sub-quadratic prover's answer.

Further, the problem underlying the PoW (to output a polynomial for a given $f\mathsf{OV}_n$ instance) is falsifiable with the added property that both the sampler *and* the verifier run in sub-quadratic time. Thus to heuristically falsify SETH, a challenger may deploy PoW challenges out into the world and, if they're often prematurely solved, we gather empirical evidence against SETH. Note that this can similarly be done for 3SUM and APSP as their polynomials can also be used for PoWs.

We note that interesting applications of heuristic falsifiability may be inherent to the intersection of average-case complexity and fine-grained complexity: One of the only other works we are aware of

that might be considered average-case fine-grained analysis, [GH16], immediately yields results that notions of heuristic falsifiability can apply to. That is, [GH16] shows that fine-grained problems related to DNA sequencing are actually *easy* when given a batch of correlated instances; thus, this analysis is average-case over a specific distribution of *correlated instances* for a fine-grained problem. This distribution for which easiness is achieved, however, is motivated to be "realistic" by the correlation of the instances attempting to match current evolutionary theory and how mutations occur within a phylogenetic tree. Thus, if a distribution over correlated instances matches well a theory of evolution *yet* [GH16]'s algorithm consistently under-performs on real-life data, this may suggest our current theory of evolution is *wrong*. Again, we can (efficiently) test a hypothesis in a concretely parameterizable way and gather evidence against it.

We find it interesting that combining average-case and fine-grained complexity seems to almost immediately bear interesting fruit in the context of heuristic falsifiability. We pose it as an open question to find more ways in which the "scientific method" can be introduced into the highly theoretical field of complexity theory so that conjectures can *tested* to give concrete parameters for our confidences for them.

7 Open Problems

To our knowledge, our work is the first to attain average-case results in the young field of conditional fine-grained complexity as described in [Wil15], and there are many ripe questions to ask of average-case fine-grained complexity. Amongst several questions regarding average-case algorithms for fine-grained problems, fine-grained cryptography, and the heuristic falsifiability of worst-case assumptions are the following:

1. Can other low-degree polynomials be found for other interesting problems in the fine-grained world to match conjectured worst-case lower bounds on them – e.g. can k-CLIQUE reduce to evaluating a family of polynomials that are both computable in time $n^{\omega k/3}$ and have degree $n^{o(1)}$?
2. Are there samplable distributions over which well-studied problems like 3SUM, OV, etc. are hard on average? Towards this, is it possible to reduce any of our polynomial families back to the problems they came from or to problems further down their reduction chain?
3. Are there any other worst-case to average-case reduction techniques that might be interesting in the fine-grained world?
4. Classically, generic derandomization from worst-case hardness assumptions proceeds in two steps: Achieving average-case hardness from worst-case hardness, and achieving strong pseudorandom generators from average-case hardness. In this paper we achieve the first step in a fine-grained way. Can pseduorandom generators be achieved in a fine-grained way from our average-case hardness?
5. Is it possible to construct a single-round coAM protocol for OV (or for $\mathcal{F}$OV) with a subquadratic time verifier (or for k-SAT with a $2^{(1-\epsilon)n}$-time verifier)?
6. Can we construct a fine-grained OWF from worst-case hardness assumptions? Is it possible to realize it from an AM protocol for $\mathcal{F}$OV as discussed in Section 5.3?
7. Standard OWFs are sufficient for secret-key cryptography as they are equivalent to Pseudorandom Generators and Pseudorandom Functions [HILL99, GGM84]. What similar equivalences hold in the fine-grained world and what fine-grained cryptography can be accomplished from

fine-grained OWFs? More generally, what standard cryptographic results translate over to fine-grained cryptography and are there any that hold only in the fine-grained world?

8. Can we heuristically falsify any of the three big worst-case fine-grained conjectures (from Section 2)? Are there other ways in which we can develop techniques for practice to influence theory and give concrete and parameterizable theoretical evidence?

Acknowledgements

We would like to thank Amir Abboud for suggesting a barrier to achieving low-degree polynomials for EDIT-DISTANCE (Section 2.2) and for pointing us to the Triangle-Collection problem in [AWY15] and a polynomial for it, Shafi Goldwasser for pointing out [Wil16] to us, Tim Roughgarden for clarifications on Remark 5, and Vinod Vaikuntanathan for comments relating to the material in Section 5.3 and Open Question 5. Thanks also to Andrej Bogdanov, Irit Dinur, Ramprasad Saptharishi, Eli-Ben Sasson, Amir Shpilka, and Amnon Ta-Shma for useful discussions.

The bulk of this work was performed while the authors were at IDC Herzliya's FACT center and supported by NSF-BSF Cyber Security and Privacy grant #2014/632, ISF grant #1255/12, and by the ERC under the EU's Seventh Framework Programme (FP/2007-2013) ERC Grant Agreement #07952. Marshall Ball is supported in part by the Defense Advanced Research Project Agency (DARPA) and Army Research Office (ARO) under Contract #W911NF-15-C-0236, and NSF grants #CNS-1445424 and #CCF-1423306. Manuel Sabin is also supported by the National Science Foundation Graduate Research Fellowship under Grant #DGE-1106400. Prashant Nalini Vasudevan is also supported by the IBM Thomas J. Watson Research Center (Agreement #4915012803).

References

[Abb17] Amir Abboud. Personal communication, 2017.

[ABW15a] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. If the current clique algorithms are optimal, so is valiant's parser. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 98–117. IEEE, 2015.

[ABW15b] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Quadratic-time hardness of LCS and other sequence similarity measures. *CoRR*, abs/1501.07053, 2015.

[AHWW15] Amir Abboud, Thomas Dueholm Hansen, Virginia Vassilevska Williams, and Ryan Williams. Simulating branching programs with edit distance and friends or: A polylog shaved is a lower bound made. *arXiv preprint arXiv:1511.06022*, 2015.

[AL13] Amir Abboud and Kevin Lewi. Exact weight subgraphs and the k-sum conjecture. In *International Colloquium on Automata, Languages, and Programming*, pages 1–12. Springer Berlin Heidelberg, 2013.

[AWW14] Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of faster alignment of sequences. In *International Colloquium on Automata, Languages, and Programming*, pages 39–51. Springer, 2014.

[AWY15] Amir Abboud, Virginia Vassilevska Williams, and Huacheng Yu. Matching triangles and basing hardness on an extremely popular conjecture. Manuscript: https://dl.dropboxusercontent.com/u/14999836/publications/MatchTria.pdf, 2015.

[Bab85] László Babai. Trading group theory for randomness. In Robert Sedgewick, editor, *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 421–429. ACM, 1985.

[BDT16] Arturs Backurs, Nishanth Dikkala, and Christos Tzamos. Tight hardness results for maximum weight rectangles. *arXiv preprint arXiv:1602.05837*, 2016.

[BGL16] Karl Bringmann, Allan Grønlund, and Kasper Green Larsen. A dichotomy for regular expression membership testing. *arXiv preprint arXiv:1611.00918*, 2016.

[BHvM09] Armin Biere, Marijn Heule, and Hans van Maaren. *Handbook of satisfiability*, volume 185. ios press, 2009.

[BI14] Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). *CoRR*, abs/1412.0348, 2014.

[BK15] Karl Bringmann and Marvin Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 79–97. IEEE, 2015.

[BK16] Andreas Björklund and Petteri Kaski. How proofs are prepared at camelot. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, pages 391–400. ACM, 2016.

[BRSV17] Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. Proofs of useful work. In submission, 2017.

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

[BT16] Arturs Backurs and Christos Tzamos. Improving viterbi is hard: Better runtimes imply faster clique algorithms. *arXiv preprint arXiv:1607.04229*, 2016.

[CFK+15] M. Cygan, F.V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer International Publishing, 2015.

[CGI+16] Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, pages 261–270, 2016.

[CIKP03] Chris Calabro, Russell Impagliazzo, Valentine Kabanets, and Ramamohan Paturi. The complexity of unique k-sat: An isolation lemma for k-cnfs. In *18th Annual IEEE Conference on Computational Complexity (Complexity 2003), 7-10 July 2003, Aarhus, Denmark*, page 135, 2003.

[CPS99] Jin-yi Cai, Aduri Pavan, and D. Sivakumar. On the hardness of permanent. In Christoph Meinel and Sophie Tison, editors, *STACS 99, 16th Annual Symposium on Theoretical Aspects of Computer Science, Trier, Germany, March 4-6, 1999, Proceedings*, volume 1563 of *Lecture Notes in Computer Science*, pages 90–99. Springer, 1999.

[CW16] Timothy M Chan and Ryan Williams. Deterministic apsp, orthogonal vectors, and more: Quickly derandomizing razborov-smolensky. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1246–1255. Society for Industrial and Applied Mathematics, 2016.

[DHW10] Holger Dell, Thore Husfeldt, and Martin Wahlén. Exponential time complexity of the permanent and the tutte polynomial. In *International Colloquium on Automata, Languages, and Programming*, pages 426–437. Springer, 2010.

[DN92] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, pages 139–147, 1992.

[DVV16] Akshay Degwekar, Vinod Vaikuntanathan, and Prashant Nalini Vasudevan. Fine-grained cryptography. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part III*, pages 533–562, 2016.

[FF90] J Feigenbaum and L Fortnow. On the random-self-reducibility of complete sets. *University of Chicago Technical Report*, pages 90–22, 1990.

[Fid72] Charles M. Fiduccia. Polynomial evaluation via the division algorithm: The fast fourier transform revisited. In Patrick C. Fischer, H. Paul Zeiger, Jeffrey D. Ullman, and Arnold L. Rosenberg, editors, *Proceedings of the 4th Annual ACM Symposium on Theory of Computing, May 1-3, 1972, Denver, Colorado, USA*, pages 88–93. ACM, 1972.

[GGH94] Mikael Goldmann, Per Grape, and Johan Håstad. On average time hierarchies. *Inf. Process. Lett.*, 49(1):15–20, 1994.

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th Annual Symposium on Foundations of Computer Science, West Palm Beach, Florida, USA, 24-26 October 1984*, pages 464–479, 1984.

[GH16] Shafi Goldwasser and Dhiraj Holden. On the fine grained complexity of polynomial time problems given correlated instances. In *Innovations in Theoretical Computer Science (ITCS)*, 2016.

[GI16] Jiawei Gao and Russell Impagliazzo. Orthogonal vectors is hard for first-order properties on sparse graphs. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:53, 2016.

[GO95] Anka Gajentaan and Mark H Overmars. On a class of $O(n^2)$ problems in computational geometry. *Computational geometry*, 5(3):165–185, 1995.

[GR17] Oded Goldreich and Guy Rothblum. Simple doubly-efficient interactive proof systems for locally-characterizable sets. Electronic Colloquium on Computational Complexity Report TR17-018, February 2017.

[GS86] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 59–68. ACM, 1986.

[GS92] Peter Gemmell and Madhu Sudan. Highly resilient correctors for polynomials. *Information processing letters*, 43(4):169–174, 1992.

[GS99] Venkatesan Guruswami and Madhu Sudan. Improved decoding of reed-solomon and algebraic-geometry codes. *IEEE Trans. Information Theory*, 45(6):1757–1767, 1999.

[Hås87] Johan Håstad. One-way permutations in NC^0. *Information Processing Letters*, 26(3):153–155, 1987.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[Hor72] Ellis Horowitz. A fast method for interpolation using preconditioning. *Information Processing Letters*, 1(4):157–163, 1972.

[IKOS08] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 433–442. ACM, 2008.

[IL89] Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, USA, 30 October - 1 November 1989*, pages 230–235, 1989.

[Imp95] R. Impagliazzo. A personal view of average-case complexity. In *Proceedings of the 10th Annual Structure in Complexity Theory Conference (SCT'95)*, SCT '95, pages 134–, Washington, DC, USA, 1995. IEEE Computer Society.

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

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

[LO87] Jeffrey C Lagarias and Andrew M. Odlyzko. Computing π (x): An analytic method. *Journal of Algorithms*, 8(2):173–191, 1987.

[Mau92] Ueli M Maurer. Conditionally-perfect secrecy and a provably-secure randomized cipher. *Journal of Cryptology*, 5(1):53–66, 1992.

[Mer78] Ralph C. Merkle. Secure communications over insecure channels. *Commun. ACM*, 21(4):294–299, 1978.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[Nao03] Moni Naor. On cryptographic assumptions and challenges. In *Annual International Cryptology Conference*, pages 96–109. Springer, 2003.

[NP85] Jaroslav Nešetřil and Svatopluk Poljak. On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae*, 26(2):415–419, 1985.

[PǏ10] Mihai Pătraşcu. Towards polynomial lower bounds for dynamic problems. In *Proceedings of the Forty-second ACM Symposium on Theory of Computing*, STOC '10, pages 603–610, New York, NY, USA, 2010. ACM.

[RR00] Ron M. Roth and Gitit Ruckenstein. Efficient decoding of reed-solomon codes beyond half the minimum distance. *IEEE Trans. Information Theory*, 46(1):246–257, 2000.

[RZ04] Liam Roditty and Uri Zwick. On dynamic shortest paths problems. In *European Symposium on Algorithms*, pages 580–591. Springer, 2004.

[Sud97] Madhu Sudan. Decoding of reed solomon codes beyond the error-correction bound. *J. Complexity*, 13(1):180–193, 1997.

[VV85] Leslie G. Valiant and Vijay V. Vazirani. NP is as easy as detecting unique solutions. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 458–463, 1985.

[VW09] Virginia Vassilevska and Ryan Williams. Finding, minimizing, and counting weighted subgraphs. In *In Proceedings of the Fourty-First Annual ACM Symposium on the Theory of Computing*, pages 455–464, 2009.

[Wil05] Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2-3):357–365, 2005.

[Wil15] Virginia Vassilevska Williams. Hardness of easy problems: Basing hardness on popular conjectures such as the strong exponential time hypothesis. In *Proc. International Symposium on Parameterized and Exact Computation*, pages 16–28, 2015.

[Wil16] Ryan Williams. Strong ETH breaks with merlin and arthur: Short non-interactive proofs of batch evaluation. In *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, pages 2:1–2:17, 2016.

[WW10] Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, 00(undefined):645–654, 2010.

A Evaluating Low Degree Polynomials

Here we recall and prove Lemma 1.

Lemma 1. *Consider positive integers N, D, and p, and an $\varepsilon \in (0, 1/3)$ such that $D > 9$, p is prime and $p > 12D$. Suppose that for some polynomial $f : \mathbb{F}_p^N \to \mathbb{F}_p$ of degree D, there is an algorithm A running in time t such that:*

$$\Pr_{\boldsymbol{x} \leftarrow \mathbb{F}_p^N} [A(\boldsymbol{x}) = f(\boldsymbol{x})] \geq 1 - \varepsilon$$

Then there is a randomized algorithm B that runs in time $O(ND^2 \log^2 p + D^3 + tD)$ such that for any $\boldsymbol{x} \in \mathbb{F}_p^N$:

$$\Pr [B(\boldsymbol{x}) = f(\boldsymbol{x})] \geq \frac{2}{3}$$

Proof. The algorithm B works as follows on input $\boldsymbol{x}$:

1. Draw two random points $\boldsymbol{y}_1, \boldsymbol{y}_2 \in \mathbb{F}_p^N$ and define the curve $\boldsymbol{c}(w) = \boldsymbol{x} + w\boldsymbol{y}_1 + w^2\boldsymbol{y}_2$ for $w \in \mathbb{F}_p$.
2. For a value of m ($< p$) to be determined later, compute $(A(\boldsymbol{c}(1)), \ldots, A(\boldsymbol{c}(m)))$ to get $(z_1, \ldots, z_m) \in \mathbb{F}_p$.
3. Run Berlekamp-Welch on $(z_1, \ldots, z_m)$. If it succeeds and outputs a polynomial $\hat{g}$, output $\hat{g}(0)$. Otherwise output 0.

To see why the above algorithm works, define the polynomial $g(w) = f(\boldsymbol{c}(w))$. g is a polynomial of degree $2D$ over the single variable w, with the property that $g(0) = f(\boldsymbol{x})$. So if we had $(2D+1)$ evaluations of g at different points in $\mathbb{F}_p$, we could retrieve g and compute $f(\boldsymbol{x})$. While we may not be able to obtain these evaluations directly (since they involve computing f), we do have access to A, which promises to be correct about the value of f on a random point with probability $2/3$.

So we replace the values $g(w) = f(\boldsymbol{c}(w))$ with $A(\boldsymbol{c}(w))$, which will hopefully be correct at several points. Now our problem is to retrieve g given a set of its alleged evaluations at m points, some of which may be wrong.

We do this by interpreting $(g(1), g(2), \ldots, g(m))$ as a Reed-Solomon encoding of g and running its decoding algorithm on $(A(\boldsymbol{c}(1)), \ldots, A(\boldsymbol{c}(m)))$, which now corresponds to a corrupt codeword. The Berlekamp-Welch algorithm can do this as long as less than $(m - 2D)/2$ of the m values of $A(\boldsymbol{c}(w))$ are wrong. We now bound the probability of too many of these values being wrong.

Let Q_w be an indicator variable such that $Q_w = 1$ if and only if $A(\boldsymbol{c}(w)) \neq f(\boldsymbol{c}(w))$. Let $Q = \sum_{w=1}^{m} Q_w$. We note at this point the fact that over the randomness of $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$, the distributions of $\boldsymbol{c}(w)$ and $\boldsymbol{c}(w')$ for any two distinct $w, w' \in \mathbb{F}_p$ are uniform and independent. This gives us the following statistics:

$$\begin{aligned} \mathrm{E}[Q] &= \varepsilon m \\ \mathrm{Var}[Q] &= m\varepsilon(1-\varepsilon) \end{aligned}$$

Thus, by the Chebyshev inequality, we have that the probability that more than a δ ($> \varepsilon$) fraction of $A(\boldsymbol{c}(w))$'s disagree with $f(\boldsymbol{c}(w))$ is:

$$\begin{aligned} \Pr\left[Q > \delta m\right] &\leq \Pr\left[|Q - \varepsilon m| > (\delta - \varepsilon)m\right] \\ &\leq \frac{\varepsilon(1-\varepsilon)}{(\delta-\varepsilon)^2 m} \leq \frac{1}{4(\delta-\varepsilon)^2 m} \end{aligned}$$

We are interested in $\delta = \frac{m-2D}{2m} = \frac{1}{2} - \frac{D}{m}$. So if we set, for instance, $m = 12D$, the bound on the probability above is at most $1/3$ if $D > 9$ and $\varepsilon < 1/3$.

So except with this small probability, the decoding algorithm correctly recovers g as $\hat{g}$, and consequently B computes $f(\boldsymbol{x})$ correctly.

Generating each $\boldsymbol{c}(w)$ and running A on it takes $O(ND\log^2 p + t)$ time. The Berlekamp-Welch algorithm then takes $O(m^3)$ time, and the final evaluation of $\widehat{g}$ takes $O(D\log^2 p)$. Hence, given A with the above properties, the running time of B is:

$$O(m(ND\log^2 p + t) + m^3 + D\log^2 p) = O(ND^2\log^2 p + D^3 + tD)$$

□

B Polynomials Computing Sums

In this section we write down polynomials that compute the bits of the sum of a pair of numbers that are given to them in bits. Without loss of generality, we will represent numbers in the two's complement form. In both cases where such representations are required (3SUM and ZWT), there are apriori bounds on the sizes of the numbers that come up – these are either numbers in the input or sums of pairs of these numbers. So we can assume that we always have enough bits to be able to represent these numbers. Under this assumption, addition in the two's complement representation is the same as adding unsigned numbers in the standard place-value representation (and ignoring the final carry). So we will present the polynomials $\{s_\ell\}_{\ell \in [d]}$ that correspond to unsigned addition (and are easier to describe) and these are the polynomials that will be used.

We label the bits of a d-bit number from 1 to d starting from the least significant bit. We then translate the semantics of the ripple-carry adder into polynomials. The polynomial $s_1 : \mathbb{F}_p^{2d} \to \mathbb{F}_p$ corresponding to the first bit of the sum is:

$$s_1(x, y) = x_1(1 - y_1) + (1 - x_1)y_1$$

The carry from this operation is given by the following polynomial:

$$c_1(x, y) = x_1 y_1$$

For every other ℓ, this pair of polynomials can be computed from earlier polynomials and the inputs as follows (hiding the arguments x and y for convenience):

$$s_\ell = (1 - x_\ell)(1 - y_\ell)c_{\ell-1} + (1 - x_\ell)y_\ell(1 - c_{\ell-1}) + x_\ell(1 - y_\ell)(1 - c_{\ell-1}) + x_\ell y_\ell c_{\ell-1}$$
$$c_\ell = x_\ell y_\ell(1 - c_{\ell-1}) + x_\ell(1 - y_\ell)c_{\ell-1} + (1 - x_\ell)y_\ell c_{\ell-1} + x_\ell y_\ell c_{\ell-1}$$

It can now be seen that $deg(s_\ell) = deg(c_\ell) = deg(c_{\ell-1}) + 2$. Along with the fact that $deg(c_1) = 2$, this implies that $deg(s_\ell) = 2\ell \leq 2d$.

These polynomials can also be computed very easily by evaluating them in order. Given $c_{\ell-1}$, both s_ℓ and c_ℓ take only a constant number of operations to compute. Hence all the s_ℓ's can be computed is $O(d \log^2 p)$ time.

C CONVOLUTION-3SUM

Here we give another average-case fine-grained hard problem based on the hardness of 3SUM. While Section 3.2 already has a polynomial whose average-case hardness is based on the 3SUM conjecture, we include this one for completeness as it is possible that either is independently hard even if the other is shown to be easy. We first recall the CONVOLUTION-3SUM (C3SUM) problem introduced by [Pǐ10] where it was shown that 3SUM has a (randomized) fine-grained reduction to C3SUM, thus allowing us to restrict our attention to it.

- C3SUM*:* Determine whether, when given three n-element arrays, A, B, and C, with entries in $\{-n^3, \ldots, n^3\}$, there exist $i, j \in [n]$ such that $A[i] + B[j] = C[i + j]$.

We now define a family of polynomials that can count solutions to a C3SUM instance (when given in binary) and refer the reader to Section 3.2 for all notation and discussion due to its similarity to $\mathcal{F}$ZWT.

For any n, let $p(n)$ denote the smallest prime number larger than n^2 and let $d = \lceil 3 \log n \rceil + 3$. We define the polynomial $f\mathsf{3SUM}_n : \mathbb{F}_p^{3nd} \to \mathbb{F}_p$ as taking in sets A, B, and C of nd variables each

and where we split each set into n groups of d variables – e.g. $A[i]$ is the i^{th} group of d variables of the nd variables in A.

$$f\mathsf{3SUM}_n(A,B,C) = \sum_{i,j\in[n]} \prod_{\ell\in[d]} \left(1 - \left(s_\ell\left(A[i],B[j]\right) - C[i+j]_\ell\right)^2\right)$$

From the same arguments in Section 3.2, we get the following theorem for $\mathcal{F}\mathsf{3SUM} = \{f\mathsf{3SUM}_n\}$.

Theorem 5. *If $\mathcal{F}$3SUM can be computed in $O(n^{1+\alpha})$ time on average for some $\alpha > 0$, then* 3SUM *can be decided in $\widetilde{O}(n^{1+\alpha})$ time in the worst case.*

Corollary 7. *If* 3SUM *requires $n^{2-o(1)}$ time to decide, $\mathcal{F}$3SUM requires $n^{2-o(1)}$ time to compute on average.*

Note that if we did not use C3SUM, we would have had n^3 terms from a more naïve construction from 3SUM and thus a gap between $\mathcal{F}$3SUM's computability and its hardness. But, with our current construction having only n^2 terms, we achieve tightness where $\mathcal{F}$3SUM is quadratic-computable but sub-quadratic-hard.

D A Tighter Reduction for $\mathcal{F}$OV

We show that sub-quadratic algorithms cannot compute $f\mathsf{OV}_n$ on even a $1/\operatorname{polylog}(n)$-fraction of inputs, assuming OV is hard on the worst case. Moreover, the techniques yield a tradeoff between adversarial complexity and provable hardness: less time implies lower success probability. Similar results can be achieved for our other polynomials, but we do not present them here.

Recall that the worst-case to average-case reduction used in Section 3 (as Lemma 1) works roughly as follows for a function f. Given an input x, the reduction produces a set of inputs $y_1,\ldots,y_m$ such that $(f(x),f(y_1),\ldots,f(y_m))$ is a Reed-Solomon codeword. Then we said that if an algorithm is correct on a $(1-\delta)$ fraction of inputs, then it is correct on close to a $(1-\delta)$ fraction of the y_i's, and so only about a δ fraction of this codeword is erroneous. As long as δ is somewhat smaller than $1/2$, we can correct these errors to recover the whole codeword and hence $f(x)$. But notice that if δ is more than $1/2$, then there is no hope of correcting the codeword, and the reduction will not work.

Because of this, the approach used in Section 3 is limited in that it cannot show, for instance, that sub-quadratic algorithms cannot compute $f\mathsf{OV}_n$ on more than a $1/3$ fraction of inputs. One thing that can be done even if more than half the codeword is corrupted, however, is list decoding. And the Reed-Solomon code turns out to have rather efficient list decoding algorithms [Sud97, GS99]. This fact was used by Cai, Pavan, and Sivakumar [CPS99] to show rather strong average-case hardness results for the Permanent using its downward self-reducibility properties.

We use their techniques to prove that sub-quadratic algorithms cannot compute $f\mathsf{OV}_n$ on even a $1/\operatorname{polylog}(n)$ fraction of inputs. The primary issue that one has to deal with when using list decoding instead of decoding is that it will yield many candidate polynomials. The insight of [CPS99], building on previous work regarding enumerative counting, is that downward self-reducibility can be used to isolate the true polynomial via recursion. And $f\mathsf{OV}_n$ turns out to have the properties necessary to do this. And, although we do not show it here, the same methodology works for $f\mathsf{OV}_n^k$ and $f\mathsf{ZWT}_n$.

Before we begin, we will present a few Lemmas from the literature to make certain bounds explicit.

First, we present an inclusion-exclusion bound from [CPS99] on the polynomials consistent with a fraction of m input-output pairs, $(x_1, y_1), \ldots, (x_m, y_m)$. We include a laconic proof here with the given notation for convenience.

Lemma 3 ([CPS99])**.** *For any polynomial q over $\mathbb{F}_p$, define* $\mathrm{Graph}(q) := \{(i, q(i)) \mid i \in [p]\}$. *Let $c > 2$, $\delta/2 \in (0,1)$, and $m \leq p$ such that $m > \frac{c^2(d-1)}{\delta^2(c-2)}$ for some d. Finally, let $I \subseteq [p]$ such that $|I| = m$. Then, for any set $S = \{(i, y_i) \mid i \in I\}$, there are less than $\lceil c/\delta \rceil$ polynomials q of degree at most d that satisfy $|\mathrm{Graph}(q) \cap S| \geq m\delta/2$.*

Corollary 8. *Let S be as in Lemma 3 with $I = \{m+1, \ldots, p\}$, for any $m < p$. Then for $m > 9d/\delta^2$, there are at most $3/\delta$ polynomials, q, of degree at most d such that $|\mathrm{Graph}(q) \cap S| \geq m\delta/2$.*

Proof. Reproduced from [CPS99] for convenience; see original for exposition.

Suppose, for contradiction, that there exists at least $\lceil c/\delta \rceil$ such polynomials. Consider a subset of exactly $N = \lceil c/\delta \rceil$ such polynomials, $\mathcal{F}$. Define $\mathsf{S}_f := \{(i, f(i)) \in \mathrm{Graph}(f) \cap S\}$, for each $f \in \mathcal{F}$.

$$\begin{aligned}
m \geq \left| \bigcup_{f \in \mathcal{F}} S_f \right| &\geq \sum_{f \in \mathcal{F}} |S_f| - \sum_{f, f' \in \mathcal{F} : f \neq f'} |S_f \cap S_{f'}| \\
&\geq N \frac{m\delta}{2} - \frac{N(N-1)(d-1)}{2} \\
&> \frac{N}{2} \left(m\delta - \frac{c(d-1)}{\delta} \right) \\
&\geq \frac{c}{2\delta} \left(m\delta - \frac{c(d-1)}{\delta} \right) \\
&= \frac{cm}{2} - \frac{c^2(d-1)}{2\delta^2} \\
&= m + \frac{1}{2} \left((c-2)m - \frac{c^2(d-1)}{\delta^2} \right) > m.
\end{aligned}$$

□

Now, we give a theorem based on an efficient list-decoding algorithm, related to Sudan's, from Roth and Ruckenstein. [RR00]

Lemma 4 ([RR00])**.** *List decoding for $[n, k]$-Reed-Solomon (RS) codes over $\mathbb{F}_p$ given a code word with almost $n - \sqrt{2kn}$ errors (for $k > 5$), can be performed in*

$$O\left(n^{3/2} k^{-1/2} \log^2 n + (n-k)^2 \sqrt{n/k} + (\sqrt{nk} + \log q) n \log^2(n/k) \right)$$

operations over $\mathbb{F}_q$.

Plugging in specific parameters and using efficient list decoding, we get the following corollary which will be useful below.

Corollary 9. *For parameters $n \in \mathbb{N}$ and $\delta \in (0,1)$, list decoding for $[m, k]$-RS codes over $\mathbb{F}_p$ where $m = \Theta(d \log n/\delta^2)$, $k = \Theta(d)$, $p = O(n^2)$, and $d = \Omega(\log n)$ can be performed in time*

$$O\left(\frac{d^2 \log^{5/2} n \mathrm{Arith}(n)}{\delta^5} \right),$$

where $\mathrm{Arith}(n)$ is a time bound on arithmetic operations over prime fields size $O(n)$.

Finally, we present a more explicitly parametrized variant of $\mathcal{F}\mathsf{OV}$, $\mathcal{G}\mathsf{OV} = \{g\mathsf{OV}_{n,d,p}\}_{n,d,p\in\mathbb{Z}^3}$, where

$$g\mathsf{OV}_{n,d,p} : \mathbb{F}_p^{2nd} \to \mathbb{F}_p$$
such that
$$g\mathsf{OV}_{n,d,p}\left(\boldsymbol{U} = \begin{bmatrix} \boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{u}_n \end{bmatrix}, \boldsymbol{V} = \begin{bmatrix} \boldsymbol{v}_1 \\ \vdots \\ \boldsymbol{v}_n \end{bmatrix}\right) := \sum_{(\boldsymbol{u}_i,\boldsymbol{v}_j)\in U\times V} \prod_{\ell\in[d]} (1 - \boldsymbol{u}_{i\ell}\boldsymbol{v}_{j\ell}).$$

Theorem 6. *If there is an algorithm that runs in time $t(n,d,p)$ for $g\mathsf{OV}_{n,d,p}$ with success probability δ on the uniform distribution, then there is an algorithm that runs in time*

$$t'(n,d,p) = O(n^{1+\gamma} + td\log^2 n/\delta^2 + d^2/\delta^5 \log^{7/2} n\mathrm{Arith}(n^2))$$

for $g\mathsf{OV}_d$ with failure probility at most $\varepsilon < 4\delta\log n/d$ for any input. $\mathrm{Arith}(n)$ *is defined to be time bound on arithmetic operations over prime fields of size $O(n)$.*

Before jumping into the proof, we observe the following corollary that essentially provides a tradeoff between runtime and hardness. Moreover, it gives a tighter hardness result on algorithms allowed to run in slightly quadratic time.

Corollary 10. *Assume $t = \Omega(d/\delta^3\log^3 n)$. If OV takes time $\Omega(n^{2-o(1)})$ time to decide, then any algorithm for $\mathcal{G}\mathsf{OV}$ that runs in time t with success probility δ on the uniform distribution must obey*

$$t/\delta^2 = \Omega(n^{2-o(1)}).$$

In particular, assuming OV takes time $\Omega(n^{2-o(1)})$, any algorithm for $\mathcal{G}\mathsf{OV}$ running in time $t = O(n^{2-\varepsilon})$, cannot succeed on a $1/n^\gamma$ fraction of the instances for any γ such that $0 < \gamma < \varepsilon/2$.

Proof. Let $(U,V) \in \{0,1\}^{2n\times d}$ be an instance of boolean-valued orthogonal vectors. Now, consider splitting these lists in half, $U = (U_0,U_1)$ and $V = (V_0,V_1)$, such that $(U_a,V_b) \in \{0,1\}^{n\times d}$ for $a,b \in \{0,1\}$. Then, define the following four sub-problems:

$$A^1 = (U_0,V_0), \qquad A^2 = (U_0,V_1), \qquad A^3 = (U_1,V_0), \qquad A^4 = (U_1,V_1).$$

Notice that given solutions to $\mathcal{G}\mathsf{OV}_d$ on A^1,A^2,A^3,A^4 we can trivially construct a solution to OV_d on (U,V).

Now, draw random $B,C \in \mathbb{F}_p^{n\times d}$ and consider the following degree 4 polynomial in x:

$$D(x) = \sum_{i=1}^{4} \delta_i(x)A^i + (B + xC)\prod_{i=1}^{4}(x-i),$$

where δ_i is the unique degree 3 polynomial over $\mathbb{F}_p$ that takes value 1 at $i \in \{1,2,3,4\}$ and 0 on all other values in $\{1,2,3,4\}$. Notice that $D(i) = A^i$ for $i = 1,2,3,4$.

Let $m > 8d/\delta^2\log n$. $D(5),D(6),\ldots,D(m+4)$. By the properties of $\mathcal{A}$ and because the $D(i)$'s are pairwise independent, $\mathcal{A}(D(i)) = g\mathsf{OV}(D(i))$ for $\delta m/2$ i's with probability $> 1 - \frac{4}{\delta m} = 1 - 1/\operatorname{polylog}(n)$, by a Chebyshev bound.

Now, because $\delta m/2 > \sqrt{16dm}$, we can run the list decoding algorithm of Roth and Ruckenstein, [RR00], to get a list of all polynomials with degree $\leq 8d$ that agree with at least $\delta m/2$ of the values. By Corollary 8, there are at most $L = 3/\delta$ such polynomials.

By a counting argument, there can be at most $4d\binom{L}{2} = O(dL^2)$ points in $\mathbb{F}_p$ on which any two of the L polynomials agree. Because $p > n^2 > 4d\binom{L}{2}$, we can find such a point, j, by brute-force in $O(L \cdot dL^2 \log^3(dL^2) \log p)$ time, via batch univariate evaluation [Fid72]. Now, to identify the correct polynomial $g\mathsf{OV}(D(\cdot))$, one only needs to determine the value $g\mathsf{OV}(D(j))$. To do so, we can recursively apply the above reduction to $D(j)$ until the number of vectors, n, is constant and $g\mathsf{OV}$ can be evaluated in time $O(d \log p)$.

Because each recursive iteration cuts n in half, the depth of recursion is $\log(n)$. Additionally, because each iteration has error probability $< 4/(\delta m)$, taking a union bound over the $\log(n)$ recursive steps yields an error probability that is $\varepsilon < 4 \log n/(\delta m)$.

As noted above, we can find the prime p in time $O(n^{1+\gamma})$, for any constant $\gamma > 0$, by binary searching $\{n^2 + 1, \ldots, 2n^2\}$ with calls to [LO87]. Taking $m = 8d \log n/\delta^2$, Roth and Ruckenstein's algorithm takes time $O(d^2/\delta^5 \log^{5/2} n\mathrm{Arith}(n^2))$, by Corollary 9, in each recursive call. The brute force procedure takes time $O(d/\delta^3 \log^3(d/\delta^2) \log n)$, which is dominated by list decoding time. Reconstruction takes time $O(\log n)$ in each round, and is also dominated.

$$t' = O(n^{1+\gamma} + td \log^2 n/\delta^2 + d^2/\delta^5 \log^{7/2} n\mathrm{Arith}(n^2)),$$

with error probability $\varepsilon < 4 \log n\delta/d$.

□

E Isolating Orthogonal Vectors

In this section, we describe a randomized reduction from OV to uOV (*unique*-OV), which is the Orthogonal Vectors problem with the promise that there is at most one pair of orthogonal vectors in the given instance.

While interesting by itself, such a reduction is also relevant to the rest of our work for the following reason. Recall that the polynomial $f\mathsf{OV}_n$ is defined over the field $\mathbb{F}_p$ where $p > n^2$. The reason p had to be more than n^2 was so that $f\mathsf{OV}_n$ would count the number of orthogonal vectors when given a boolean input, and this number could be as large as n^2. If we wanted a polynomial that did this for uOV, this restriction on the characteristic of the field wouldn't exist. p would have still to be $\Omega(d)$ for the random self-reduction to work, but this is much smaller than n^2 in our setting, and this could possibly allow applications of our results that would not be viable otherwise.

Recall that an important reason for believing that there is no sub-quadratic algorithm for OV is that such an algorithm would break SETH due to a fine-grained reduction from k-SAT [Wil05]. If all one wanted was a similar reason to believe that uOV was hard, one could attempt to reduce k-SAT to uOV. A natural approach to doing so would be to first reduce k-SAT to unique-k-SAT and then apply the reduction from [Wil05], as it translates the number of satisfying assignments to the number of orthogonal vectors.

However, the isolation lemma for k-SAT due to Valiant and Vazirani [VV85] turns out to not work for this purpose because it blows up the number of variables in the k-SAT instance it operates on, and the resulting reduction would not be fine-grained enough to provide the requisite lower bounds for uOV based on SETH. One way to circumvent this is that Calabro et al. [CIKP03] provide an alternative that does preserve the number of variables and shows that SETH implies an analogous conjecture for unique-k-SAT, and this can be used in the first step of the reduction so that the reduction chain would go from k-SAT to unique-k-SAT to uOV.

We instead start with OV itself and apply techniques from [VV85] directly to it, so a reduction chain of k-SAT to OV to uOV can be achieved. Throughout this section, we will use OV_d (uOV_d)

to denote the OV (respectively uOV) problem over vectors of dimension d. We start by describing a search-to-decision reduction for OV/uOV.

Lemma 5. *If, for some $c, c' \geq 1$, there is an $(n^c d^{c'})$-time algorithm for OV_d, then there is an $O(n^c d^{c'})$-time algorithm that finds a pair of orthogonal vectors in any OV_d instance (if it exists) except with negligible probability. Further, the same is true for uOV_d.*

Proof. Let A be an algorithm for deciding OV_d that has negligible error and runs in time $n^c d^{c'}$. Given a YES instance (U, V) of OV_d, where U and V have n vectors each, our search algorithm starts by dividing U and V into halves (U_0, U_1) and (V_0, V_1), where each half has roughly $\lfloor n/2 \rfloor$ vectors. Since there was a pair of orthogonal vectors in (U, V), at least one of (U_0, V_0), (U_0, V_1), (U_1, V_0), and (U_1, V_1) must contain a pair of orthogonal vectors. Run A on all four of these to identify one that does, and recurse on that one until the instance size reduces to a constant, at which point try all pairs of vectors and find an orthogonal pair. If at some point in this process A says that none of the four sub-instances contains a pair of orthogonal vectors, or if at the end there are no orthogonal pairs, give up and fail.

Since the size of the instances reduces by a constant factor each time, the number of calls made to A is $O(\log n)$. So since A makes mistakes with negligible probability, by union bound, the whole search algorithm fails only with a negligible probability. Copying over inputs to run A on takes only linear time in the input size. Accounting for this, the running time of the algorithm is:

$$\begin{aligned} T(n) &\leq 8\left(\frac{n}{2}\right)^c d^{c'} + 8\left(\frac{n}{4}\right)^c d^{c'} + \cdots + 8 \cdot O(d^{c'}) \\ &\leq 8n^c d^{c'} \left(\sum_{k=0}^{\infty} \frac{1}{2^{ck}}\right) = O(n^c d^{c'}) \end{aligned}$$

It can be seen that the same proof goes through for uOV_d as well. □

Theorem 7. *If, for some $c, c' \geq 1$, there is an $(n^c d^{c'})$-time algorithm for $\mathsf{uOV}_{d'}$, then there is an $\widetilde{O}(n^c d^{c'})$-time algorithm for OV_d, where $d' = d + 4\log n + 2$.*

The reduction is along the lines of that from SAT to unique-SAT from [VV85], and makes use of the following lemma, which is a special case of the one used there.

Lemma 6. *Let $S \subseteq \{0,1\}^d \times \{0,1\}^d$ be a set such that $2^{k-1} \leq |S| < 2^k$ for some k. With constant probability over randomly chosen $\boldsymbol{M}_0, \boldsymbol{M}_1 \in \{0,1\}^{(k+1)\times n}$ and $\boldsymbol{b} \in \{0,1\}^{(k+1)}$, there is a unique $(\boldsymbol{x}, \boldsymbol{y}) \in S$ such that $\boldsymbol{M}_0\boldsymbol{x} + \boldsymbol{b} = \boldsymbol{M}_1\boldsymbol{y}$ (over $\mathbb{F}_2$).*

The above lemma follows from the observation that over all $\boldsymbol{M}_0$, $\boldsymbol{M}_1$ and $\boldsymbol{b}$, the set

$$\{h_{\boldsymbol{M}_0,\boldsymbol{M}_1,\boldsymbol{b}}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{M}_0\boldsymbol{x} + \boldsymbol{b} - \boldsymbol{M}_1\boldsymbol{y}\}$$

is a universal family of hash functions. We refer the reader to [VV85] for the proof.

Proof of Theorem 7. Let A be an $O(n^c d^{c'})$-time search algorithm for $\mathsf{uOV}_{d'}$ – such an algorithm exists by our hypothesis and Lemma 5. We would like to use it to decide an instance (U, V) of OV_d. What are the instances of $\mathsf{uOV}_{d'}$ that we could run A on to help us in our task?

Suppose we knew that in (U, V) there were exactly m pairs of orthogonal vectors. Let k be such that $2^{k-1} \leq m < 2^k$. Lemma 6 says that if we choose $\boldsymbol{M}_0, \boldsymbol{M}_1 \in \{0,1\}^{(k+1)\times d}$ and $\boldsymbol{b} \in \{0,1\}^{(k+1)}$ at random, then with some constant probability, there is exactly one pair of vectors $\boldsymbol{u} \in U$, $\boldsymbol{v} \in V$

such that $\langle \boldsymbol{u}, \boldsymbol{v} \rangle = 0$ and $\boldsymbol{M}_0\boldsymbol{u} + \boldsymbol{b} = \boldsymbol{M}_1\boldsymbol{v}$. If we could somehow encode the latter condition as part of the orthogonal vector problem, we could hope to get a uOV instance from (U, V).

Consider the encoding schemes E_0 and E_1 described next. For any vector $\boldsymbol{x}$, $E_0(\boldsymbol{x})$ is a vector twice as long as $\boldsymbol{x}$, where each 0 in $\boldsymbol{x}$ is replaced by "0 1" and each 1 is replaced by "1 0". $E_1(\boldsymbol{x})$ is similar, but here a 0 is replaced by "1 0" and a 1 is replaced by "0 1". The property of these encodings that make them useful for us is that $\langle E_0(\boldsymbol{x}), E_1(\boldsymbol{y}) \rangle = 0$ if and only if $\boldsymbol{x} = \boldsymbol{y}$.

Putting ideas from the above two paragraphs together, consider the process where we pick $\boldsymbol{M}_0, \boldsymbol{M}_1, \boldsymbol{b}$ at random, and to each $\boldsymbol{u} \in U$, we append the vector $E_0(\boldsymbol{M}_0\boldsymbol{u}+\boldsymbol{b})$, and to each $\boldsymbol{v} \in V$, we append $E_1(\boldsymbol{M}_1\boldsymbol{v})$. Let the entire resulting instance be (U', V').

For any $\boldsymbol{u}' \in U$ and $\boldsymbol{v}' \in V$, $\langle \boldsymbol{u}', \boldsymbol{v}' \rangle = \langle \boldsymbol{u}, \boldsymbol{v} \rangle + \langle E_0(\boldsymbol{M}_0\boldsymbol{u} + \boldsymbol{b}), E_1(\boldsymbol{M}_1\boldsymbol{v}) \rangle = 0$ if and only if $\langle \boldsymbol{u}, \boldsymbol{v} \rangle = 0$ and $\boldsymbol{M}_0\boldsymbol{u} + \boldsymbol{b} = \boldsymbol{M}_1\boldsymbol{v}$. So it follows that with some constant probability, (U', V') has a unique pair of orthogonal vectors.

Generalising slightly, if we knew that an instance (U, V) had either between 2^{k-1} and 2^k pairs of orthogonal vectors or none, then to decide which is the case, all we need to do is to do the above conversion to (U', V'), pad the vectors with 0's to get them to dimension d', and run A on it. If there were no orthogonal vectors, A can never return a valid answer, and in the other case, with a constant probability there will be a unique pair of orthogonal vectors that A will find. This can be repeated, say, $\log^2 n$ times to get a negligible probability of failure.

But we do not actually know much about the number of pairs of orthogonal vectors in an instance that is given to us. This is easy to deal with, though – simply run the above algorithm for all possible values of k, from 0 to $2 \log n$. If there are indeed some pairs of orthogonal vectors, then one of these values of k was the right one to use and the corresponding iteration would give us a pair of orthogonal vectors, except with negligible probability. If there are no orthogonal vectors, then we will never find such a pair.

Each (U', V') takes at most $O(nd \log n)$ time to prepare, and an execution of A takes $O(n^c d^{c'})$ time. This is done $\log^2 n$ times for each value of k, which is from $[2 \log n]$. So the total time taken by the above algorithm is $O(\log^3 n(nd \log n + n^c d^{c'})) = \widetilde{O}(n^c d^{c'})$. □

ECCC ISSN 1433-8092
https://eccc.weizmann.ac.il