New Techniques for Proving Fine-Grained Average-Case Hardness

Mina Dalirrooyfard[1]
CSAIL
MIT
Cambridge, MA, USA
minad@mit.edu

Andrea Lincoln[1]
CSAIL
MIT
Cambridge, MA, USA
andreali@mit.edu

Virginia Vassilevska Williams[2]
CSAIL
MIT
Cambridge, MA, USA
virgi@mit.edu

2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS) | 978-1-7281-9621-3/20/$31.00 ©2020 IEEE | DOI: 10.1109/FOCS46700.2020.00077

***Abstract*—The recent emergence of fine-grained cryptography strongly motivates developing an average-case analogue of Fine-Grained Complexity (FGC).**

Prior work [Goldreich-Rothblum 2018, Boix-Adserà et al. 2019, Ball et al. 2017] developed worst-case to average-case fine-grained reductions (WCtoACFG) for certain algebraic and counting problems over natural distributions and used them to obtain a limited set of cryptographic primitives. To obtain stronger cryptographic primitives based on standard FGC assumptions, ideally, one would like to develop WCtoACFG reductions from the core hard problems of FGC, Orthogonal Vectors (OV), CNF-SAT, 3SUM, All-Pairs Shortest Paths (APSP) and zero-k-clique. Unfortunately, it is unclear whether these problems actually are hard for any natural distribution. It is known, that e.g. OV can be solved quickly for very natural distributions [Kane-Williams 2019], and in this paper we show that even counting the number of OV pairs on average has a fast algorithm.

This paper defines new versions of OV, kSUM and zero-k-clique that are both worst-case and average-case fine-grained hard assuming the core hypotheses of FGC. We then use these as a basis for fine-grained hardness and average-case hardness of other problems. The new problems represent their inputs in a certain "factored" form. We call them "factored"-OV, "factored"-zero-k-clique and "factored"-3SUM. We show that factored-k-OV and factored kSUM are equivalent and are complete for a class of problems defined over Boolean functions. Factored zero-k-clique is also complete, for a different class of problems.

Our hard factored problems are also simple enough that we can reduce them to many other problems, e.g. to edit distance, k-LCS and versions of Max-Flow. We further consider counting variants of the factored problems and give WCtoACFG reductions for them for a natural distribution. Through FGC reductions we then get *average-case* hardness for well-studied problems like regular expression matching from standard *worst-case* FGC assumptions.

To obtain our WCtoACFG reductions, we formalize the framework of [Boix-Adserà et al. 2019] that was used to give a WCtoACFG reduction for counting k-cliques. We define an explicit property of problems such that if a problem has that property one can use the framework on the problem to get a WCtoACFG self reduction. We then use the framework to slightly extend Boix-Adserà et al.'s average-case counting k-cliques result to average-case hardness for counting arbitrary subgraph patterns of constant size in k-partite graphs.

The fine-grained public-key encryption scheme of [LaVigne et al.'20] is based on an average-case hardness hypothesis for the *decision* problem, zero-k-clique, and the known techniques for building such schemes break down for algebraic/counting problems. Meanwhile, the WCtoACFG reductions so far have only been for counting problems. To bridge this gap, we show that for a natural distribution, an algorithm that detects a zero-k-clique with high enough probability also implies an algorithm that can count zero-k-cliques with high probability. This gives hope that the FGC cryptoscheme of [LaVigne et al.'20] can be based on standard FGC assumptions.

***Keywords*-Fine-Grained Complexity, Average-Case Lower Bounds, Worst-Case to Average-Case**

1 2

I. Introduction

Fine-grained complexity (FGC) is an active research area that seeks to understand why many problems of interest have particular running time bounds $t(n)$ that are easy to achieve with known techniques, but have not been improved upon significantly in decades, except by $t(n)^{o(1)}$ factors. FGC has produced a versatile set of tools that have resulted in surprising *fine-grained* reductions that together with popular hardness hypotheses explain the running time bottlenecks for a large variety of problems [Vas18]. The reductions of FGC have, for example, explained the difficulty of improving over the $n^{2-o(1)}$ time algorithms for Longest Common Subsequence (LCS) by giving a tight reduction from k-SAT, and thus showing that an improved LCS algorithm would violate the Strong Exponential Time Hypothesis (SETH) [ABV15].

There are three main problems, with associated hardness hypotheses about their running times, that FGC

[1]NSF CAREER Award, NSF Grants CCF-1740519 and CCF-1909429

[2]Supported by an NSF CAREER Award, NSF Grants CCF-1740519 and CCF-1909429, a BSF Grant BSF:2012338, a Google Research Fellowship and a Sloan Research Fellowship.

2575-8454/20/$31.00 ©2020 IEEE
DOI 10.1109/FOCS46700.2020.00077

primarily uses as sources of hardness reductions (see [Vas18]). The three core hard problems are All Pairs Shortest Paths (APSP), hypothesized to require $n^{3-o(1)}$ time in n-node graphs[3], the 3SUM problem, hypothesized to require $n^{2-o(1)}$ time on n integer inputs, and the Orthogonal Vectors (OV) problem, hypothesized to require $n^{2-o(1)}$ time for n vector inputs of dimension $\omega(\log n)$ (the OV hypothesis is implied by SETH [Wil07]).

While it is unknown whether these three hypotheses are equivalent, some work suggests they might not be [CGI+16]. There is a problem, Zero Triangle, on n node graphs that requires $n^{3-o(1)}$ time under both the 3SUM and the APSP hypothesis [VW18], [VW13]. Zero Triangle asks if an n node graph with integer edge weights contains a triangle whose three edge weights sum to 0. A natural extension of Zero Triangle, zero-k-clique (where one wants to detect a k-clique with edge weight sum 0), is conjectured to require $n^{k-o(1)}$ time. There are also some simple to define problems on n node graphs that require $n^{3-o(1)}$ time under three core hardness hypotheses (SETH, APSP and 3SUM): Matching Triangles and Triangle Collection [AVY18].

Recently there has been increased interest in developing average-case fine-grained complexity (ACFGC), with a new type of *fine-grained cryptography* as a main motivation [BRSV17], [BRSV18], [GR18], [LLV19], [BBB19]. The main goal is to identify a problem P that requires some $t(n)^{1-o(1)}$ time on average for an easily sampled distribution, and then to build interesting cryptographic primitives from this problem, where any honest party only needs to run a very fast algorithm, in some $t'(n) \leq O(t(n)^c)$ time for c much smaller than 1, while an adversary would need to run at least in $t(n)^{1-o(1)}$ time, unless problem P can be solved fast on average.

To obtain average-case fine-grained hard problems, one would like to be able to obtain worst-case to average-case fine-grained reductions for natural problems that are hypothesized to be fine-grained hard in the worst-case[4]. This is what prior work does.

The problems for which fine-grained worst-case to average-case hardness reductions are known are mostly algebraic or counting problems, such as counting k-cliques [GR20], [GR18], [BRSV18], [BBB19], or some problems involving polynomials. Some limited cryptographic primitives have been obtained from such problems, e.g. fine-grained proofs-of-work [BRSV18], [BRSV17]. Building fine-grained one-way functions or fine-grained public key cryptography based on any worst-case FGC hardness assumption is still an open problem. Such primitives have been developed, based on plausible assumptions about the average-case complexity of zero-k-clique [LLV19]. This motivates the following question: *Is there a fine-grained worst-case to average-case reduction for zero-k-clique?*

As prior work showed worst-case to average-case case reductions for counting cliques, a natural approach to obtaining worst-case to average-case reductions for the detection variant of zero-k-clique is to give a fine-grained reduction from counting to decision. A tight reduction is not known for the worst-case version of the problem. It turns out that a fine-grained reduction from counting to decision for zero-k-clique is possible in the average-case for a natural distribution with certain parameters, if the detection probability is high enough. We prove this in the full version [DLW20]. While the parameters are currently not good enough to imply a worst-case to average-case reduction for (the decision version of) zero-k-clique, the reduction gives hope that the fine-grained public-key scheme of [LLV19] can eventually be based on a standard FGC (worst-case) hardness assumption.

The next natural question is whether worst-case to average-case reductions are possible for the other core problems of FGC, and in particular for OV (as it is as far as we know unrelated to zero-k-clique). Consider the most natural distribution for OV: given a fixed probability $p \in (0,1)$, one generates n vectors of dimension $d = \omega(\log n)$ by selecting for each vector v and $i \in [d]$ independently, v_i to be 1 with probability p and 0 otherwise. Kane and Williams [KW19] showed that for every p, there is an $\varepsilon_p > 0$ and an $O(n^{2-\varepsilon_p})$ time algorithm that solves OV on instances generated from the above distribution with high probability. Thus, for this distribution (if the OV conjecture is true), there can't be a fine-grained (n^2, n^2)-worst-case to average-case reduction for OV. In the full paper [DLW20] we also show that even the counting version of OV, in which one wants to determine the number of pairs of orthogonal vectors, has a truly-subquadratic time algorithm that works with high probability over the same distribution. Thus, even counting OV cannot be average-case $n^{2-o(1)}$-hard. (Though, it could be fine-grained average-case hard for a different time function. We leave this to future work.)

[3] All hypotheses are for the word-RAM model of computation with $O(\log n)$ bit words.

[4] Well, even more ideally, one would like to use problems that are provably unconditionally average-case hard, such as the problems from the known time-hierarchy theorems, but these problems are difficult to work with and there are no known techniques to build cryptography from them.

The first key contribution of this paper is in defining a new type of problem, a "factored problem" that is fine-grained hard from a core FGC assumption, whose counting version is average-case hard for a natural distribution again under a core FGC assumption, and that is also simple enough so that one can reduce it to well-studied problems and develop average-case hardness for them.

While developing worst-case to average-case reductions for our factored problems, we formalize the worst-case to average-case fine-grained reductions framework of Boix et al. [BBB19]. We identify a property of problems (the existence of a "good polynomial") that makes it possible for these problems to have such a worst-case to average-case reduction. Originally, [BBB19] gave average-case hardness for counting k-Cliques in Erdös-Renyi graphs using their framework. Along the way of generalizing their framework, we also obtain a worst-case to average-case reduction for counting copies of H for any k-node H, where the distribution for the average-case instance is again for Erdö-Renyi graphs. We achieve this using a new technique we call Inclusion-Edgesclusion.

In the rest of the introduction we will present our results mentioned in the above two paragraphs.

A. The factored problems

We call the problems we introduce "factored problems" (a full formal definition is in Section II). To define them, let us first define a *factored vector*. Let b and g be positive integers. A (g,b)-factored vector, v, is made up of g sets $v[1],\ldots,v[g]$. Each set is a subset $v[i] \subseteq \{0,1\}^b$. Roughly speaking, a factored vector v represents many $b \cdot g$ binary vectors, namely a concatenation $x_1,x_2,\ldots,x_g$ for each choice of a g-tuple of vectors $x_i \in v[i]$ for all i. For example, for $g=2$ and $b=3$, let v be a factored vector where $v[0]=\{001,010\}$ and $v[1]=\{010,110\}$. A natural interpretation of v is that it is a set of the following 4 binary vectors, by concatenating each member of $v[0]$ with each member of $v[1]$, that is $\{001010,001110,010010,010110\}$.

Now, consider a function f that takes a $2b$-bit input $x_1,\ldots,x_b,y_1,\ldots,y_b$ and returns a value in $\{0,1\}$; we can consider f as a Boolean function. Then, for two factored vectors v and v' and a coordinate $i \in [g]$, we can consider the number of pairs of b-bit vectors $x \in v[i], y \in v'[i]$ that f accepts. This is $accept_f(v,v',i) := \sum_{x \in v[i], y \in v'[i]} f(x_1,\ldots,x_b,y_1,\ldots,y_b)$, where $x = x_1 \ldots x_b$ and $y = y_1 \ldots y_b$. If we take the product $\prod_{i=1}^{g} accept_f(v,v',i)$, we would obtain the number of pairs of $b \cdot g$-length vectors represented by v and v' that are accepted by f, where f is said to accept a pair of $b \cdot g$-length vectors if it accepts each of the g pairs of chunks of b-length subvectors between positions $(i-1)b+1$ to ib for $i \in [g]$.

Then we can define the factored problem for f, F2-f that given two sets S and T of n (g,b)-factored vectors, computes the sum $\sum_{v \in S, v' \in T} \prod_{i=1}^{g} accept_f(v,v',i)$, i.e. the total number of pairs of vectors represented by vectors in S and T that are accepted by f. For technical reasons, we restrict the values $g = o(\lg(n)/\lg\lg(n))$ and $b = o(\lg(n))$, so that each factored vector can be represented with at most $gb2^b$ bits (g sets of at most 2^b vectors of length b).

Depending on the function f, we get different versions of a factored problem. If f on b-length vectors x and y, returns 1 iff $x \cdot y = 0$, then we get the factored OV problem F2-OV. If f returns 1 if the XOR of x and y is 0, we get the F2-XOR problem, and if f returns 1 iff $x+y=0$ when viewed as integers, we get the F2-SUM problem.

More generally, f can be defined over $k \cdot b$-length vectors, for integer $k \geq 2$, taking k-tuples of b-length binary vectors to $\{0,1\}$. Then analogously we can define Fk-f to compute the number of k-tuples of vectors represented by some k-tuple of factored vectors, one from each n-sized input set S_i, $i \in [k]$, so that f accepts the k-tuple. This way we can define Fk-OV, Fk-XOR, Fk-SUM etc, the factored versions of k-OV, k-XOR and k-SUM.

Similarly to these problems defined on k-tuples of sets of factored vectors, we define problems reminiscent to k-clique. Here f is a function that takes $\binom{k}{2}$-tuples of b-length vectors to $\{0,1\}$, one is given a graph whose edges are labeled by factored vectors and the factored f k-clique problem, F$f$$k$C, asks to compute the number of $\binom{k}{2}$-tuples of vectors that are accepted by f and are represented by the factored vectors labeling the edges of a k-clique in the graph. We focus in particular on the factored zero-k-clique problem, FZkC, in which f corresponds to returning whether the sum of $\binom{k}{2}$ b-bit numbers is 0.

B. Results for factored problems

We will summarize the results around our factored problems below. They appear in the full version [DLW20]. We give a visual summary of our results in Figure 1. We use the shortened names for many of the problems in the figure. The results will concern both counting and decision versions of our factored problems. The decision versions ask whether the count is nonzero, whereas the counting versions ask for the exact count. When we want the counting version, we

will place # in front of the name of the problem. See the Preliminaries (Section II) for more details.

Summary.: We first provide an overview summary of our results.

First we show that the factored versions of k-OV, k-SUM and k-XOR are all $n^{k-o(1)}$-fine-grained hard under SETH. We also show that the factored version of zero-3-clique (FZ3C) is $n^{3-o(1)}$-fine-grained hard based on any of the three core hypotheses of FGC (SETH, or the APSP or 3-SUM hypothesis). Additionally, we show that the counting versions of these factored problems are as hard in their natural uniform average-case as they are in the worst case. Moreover, we show that many natural problems, like counting regular expression matchings, reduce from our factored problems. This even implies fine-grained average-case hardness for these problems over some explicit distributions.

Thus our factored problems do three things simultaneously:

- Instead of trying to use the uniform average-case of the core problems of FGC as central problems in a network of average-case reductions, we can use the factored versions of the core problems in FGC. For example, the counting variant of factored OV (#F2-OV) is hard in its uniform average case from the worst-case OV hypothesis. Generically, our factored problems serve as an alternative central problem for average-case hardness. To demonstrate this, we give reductions from counting factored problems to four problems in graph algorithms and sequence alignment (including counting regular expression matchings).
- The factored versions of the core problems are sufficiently expressive that they are complete for the large class of factored problems. In particular, Fk-OV, Fk-XOR, and Fk-SUM are complete for the class of problems of the form Fk-$\mathfrak{f}$ over all f, while FZkC is complete for the class of problems F$\mathfrak{f}k$C over all f. Despite this expressiveness we are still able to reduce our factored problems to many natural problems. We give fine-grained reductions from our factored problems to k-LCS, Edit Distance and a labeled version of Max Flow.
- Abboud et al. [AVY18] gave two problems, Triangle Collection and Matching Triangles that are hard from all three core assumptions in FGC. They also showed that one can reduce Triangle Collection [5] to several natural problems in graph algorithms. Unfortunately, however, neither Triangle Collection, nor Matching Triangles are known to be hard on average. One of our factored problems, FZ3C is also hard from all three core assumptions. Moreover, the counting version of FZ3C is additionally $n^{3-o(1)}$ hard in the average-case from all three core assumptions of FGC. Thus, problems that reduce from counting FZ3C get average-case hardness for some explicit average-case distribution. We give two examples of problems that reduce from counting FZ3C.Hence if you are interested in average-case hardness then counting FZ3C might be a better source for reductions than, say Matching Triangles or Triangle Collection.

[5] Actually a version of the problem that is still hard under all three assumptions.

Fine-grained hardness for factored problems: Here we show that our factored problems are fine-grained hard under standard FGC hypotheses.

We first show that a single call to a factored problem solves its non-factored counterpoint.

Theorem I.1. *In $O(n)$ time, one can reduce an instance of size n of k-OV, k-XOR, k-SUM and ZkC to a single call to an instance of size $\tilde{O}(n)$ of Fk-OV, Fk-XOR, Fk-SUM and FZkC, respectively.*

The above theorem holds both in the decision and counting context. It gives fine-grained hardness for the factored variants of all our problems, under the hypothesis that the original variants are hard. Note that k-XOR, k-SUM have $\tilde{O}(n^{\lceil k/2 \rceil})$ time algorithms. However, we have $n^{k-o(1)}$ conditional lower bounds for all of Fk-OV, Fk-XOR, Fk-SUM and FZkC. So, while we do get fine-grained hardness from the k-XOR and k-SUM hypotheses, this hardness is not tight. The hardness is tight from the k-OV and ZkC hypotheses however.

Now we give fine-grained hardness for FZ3C under all three core hypotheses from FGC.

Theorem I.2. *If FZ3C (even for $b = o(\log n)$ and $g = o(\log(n)/\log\log(n))$) can be solved in $O(n^{3-\varepsilon})$ time for some constant $\varepsilon > 0$, then SETH is false, and there exists a constant $\varepsilon' > 0$ such that 3-SUM can be solved in $O(n^{2-\varepsilon'})$ time and APSP can be solved in $O(n^{3-\varepsilon'})$ time.*

Worst-case to average-case reductions for factored problems: We show that our factored problems admit fine-grained worst-case to average-case reductions. Our first theorem about this is a worst-case to average-case fine-grained reduction for the counting version of Fk-$\mathfrak{f}$ for a natural distribution (defined in Definition 18). The proof appears in the full version [DLW20].

Theorem I.3. *Let μ be a constant such that $0 < \mu < 1$. Suppose that average-case #Fk-$\mathfrak{f}^{\mu}$ (see definition 18,*

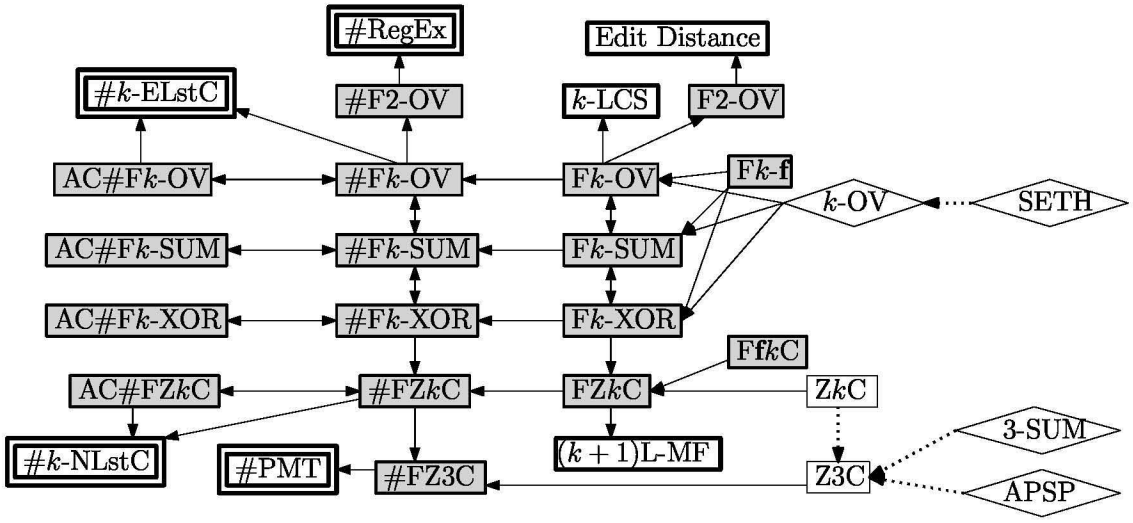

Figure 1: A summary of the reductions to and from factored problems in the paper. The problems in diamonds are the core problems of FGC. The full lines are reductions from this paper, while doted lines are pre-existing reductions. The problems in gray boxes are our factored problems. The problems in thick lined boxes are the problems we reduce from factored problems. For the problems surrounded by a thick-lined double box we have generated an explicit average case distributions on which they are hard (but it is not the uniform distribution). These results appear in the full version [DLW20].

this is an iid distribution which has ones with probability μ) can be solved in time $T(n)$ with probability at least $1-1/(lg(n)^{kg}\lg\lg(n)^{kg})$. Then worst-case #Fk-f can be solved in time $\tilde{O}(T(n))$ [6].

When $\mu = 1/2$ average-case #Fk-f$^{\mu}$ is average-case #Fk-f.

Thus, if we have worst-case fine-grained hardness for #Fk-f for some f, then we get average-case hardness for the same problem over a natural distribution. In particular, in the corollary below we obtain average-case hardness for #Fk-OV, # Fk-SUM , #Fk-XOR, based on the standard FGC hardness of k-OV, k-SUM, k-XOR (as implied by Theorem I.1).

Corollary I.4. *If average-case #Fk-OV can be solved in time $T(n)$ with probability $1-1/(lg(n)^{gk}\lg\lg(n)^{gk})$ then worst-case #Fk-OV can be solved in time $\tilde{O}(T(n))$* [6].

If average-case # Fk-SUM can be solved in time $T(n)$ with probability $1-1/(lg(n)^{gk}\lg\lg(n)^{gk})$ then worst-case # Fk-SUM can be solved in time $\tilde{O}(T(n))$ [6].

If average-case #Fk-XOR can be solved in time $T(n)$ with probability $1-1/(lg(n)^{gk}\lg\lg(n)^{gk})$ then worst-case #Fk-XOR can be solved in time $\tilde{O}(T(n))$ [6].

[6]Note that given that $g = o(\lg(n)/\lg\lg(n))$ then a probability of $1-1/n^{\varepsilon}$ will be high enough for any $\varepsilon > 0$.

Similarly, we obtain fine-grained average-case hardness for #FfkC, based on the fine-grained worst-case hardness of #FfkC.

Theorem I.5. *Let μ be a constant and $0 < \mu < 1$. If average-case #FfkC^{μ} (see Definition 18, this is an iid distribution which has ones with probability μ) can be solved in time $T(n)$ with probability $1-1/(lg(n)^{k^2g}\lg\lg(n)^{k^2g})$ then worst-case #FfkC can be solved in time $\tilde{O}(T(n))$* [6].

When $\mu = 1/2$ average-case #FfkC^{μ} is average-case #FfkC.

By Theorem I.5, we have the following result for #FZkC in particular.

Corollary I.6. *If average-case #FZkC can be solved in time $T(n)$ with probability $1-1/(lg(n)^{k^2g}\lg\lg(n)^{k^2g})$ then worst-case #FZkC can be solved in time $\tilde{O}(T(n))$* [6].

Thus in particular we obtain fine-grained average-case hardness for counting factored zero-3-cliques, based on the hardness of zero-3-clique, and thus based on the APSP and 3-SUM hypotheses.

Completeness for Fk-OV, Fk-SUM , Fk-XOR and FZkC: Let $k \geq 2$ be a fixed integer. Consider the class of problems Fk-f defined over all boolean functions f on kb-length inputs. Our first sequence of results show

that Fk-OV, Fk-SUM and Fk-XOR are complete for the class, so that a $T(n)$ time algorithm for any of these problems would imply an $\tilde{O}(T(n))$ time algorithm for Fk-f for any f.

To prove this, we first show that Fk-XOR is complete for the class:

Theorem I.7. *If we can solve #Fk-XOR with g sets of k^3b length vectors in time $T(n)$ then, for any f, we can solve a #Fk-f instance with g sets of b length vectors in time $T(n)+\tilde{O}(n)$ time.*

We then show that Fk-OV, Fk-SUM and Fk-XOR are equivalent.

Theorem I.8. *If any of #Fk-OV, # Fk-SUM , or #Fk-XOR can be solved in $T(n)$ time then all of #Fk-OV, # Fk-SUM , and #Fk-XOR can be solved in $\tilde{O}(T(n))$ time.*

The above two theorems imply the final completeness theorem:

Theorem I.9. *If any of #Fk-OV, # Fk-SUM , or #Fk-XOR can be solved in $T(n)$ time then #Fk-f can be solved in $\tilde{O}(T(n))$ time.*

We also consider the class of problems (#)Ffk$C defined by Boolean functions f on $\binom{k}{2}b$-length inputs. We show that (#)FZkC is complete for this class.

Theorem I.10. *If (#)FZkC can be solved in $T(n)$ time then (#)Ff kC for any f, can be solved in $\tilde{O}(T(n)+n^2)$ time.*

Thus our factored problems corresponding to core problems in FGC, are the hard problems for natural classes of factored problems.

Fine-grained hardness for well-studied problems, based on the hardness of factored problems: The results we mention here appear in the full version [DLW20]. The main upshot is that the factored problems are both hard and also simple enough to imply hardness for basic problems in graph and string algorithms. Some of the results are based on the hardness of FZ3C which implies hardness from all of SETH, 3-SUM and APSP. Some come from Fk-f which implies hardness from SETH.

Partitioned Matching Triangles. First we define the *Partitioned Matching Triangles* problem (PMT) as follows: Given g disjoint n-node graphs with node colors, is there a triple of colors a,b,c so that every one of the g graphs contains a triangle whose nodes are colored by a,b,c? The counting variant of PMT is to count the total number of such g-tuples of colored triangles.

Abboud et al. [AVY18] consider the related Matching Triangles problem mentioned earlier in the introduction, and show that it is hard from all three core FGC hypotheses. In the Matching Triangles problem one is given an integer T and a node-colored graph G and one wants to know if there is a triple of colors a,b,c so that there are at least T triangles in G colored by a,b,c.

We observe first that for the particular parameters for which Matching Triangles is shown to be hard in [AVY18], one can actually reduce Matching Triangles in a fine-grained way to Partitioned Matching Triangles (PMT), so that the latter problem is also hard from all three hypothesis. Furthermore, we give a powerful reduction to PMT from FZ3C. Moreover, our reduction also holds between the counting versions of the problems, so that we get fine-grained *average-case* hardness for counting PMT under all three hypotheses as well.

Theorem I.11. *If (#)Partitioned Matching Triangles can be solved in $T(n)$ time, then we can solve (#)FZ3C in time $\tilde{O}(T(n)+n^2)$.*

k-color Node Labeled st Connectivity. In the k-color Node Labeled *st* Connectivity Problem (k-NLstC) one is given an acyclic graph $G=(V,E)$ with two designated nodes $s,t\in V$, and colors on all nodes in $V\setminus\{s,t\}$ from a set of colors C. One is then asked whether there is a path from s to t in G using at most k node colors.

We give a fine-grained reduction from FZkC to k-NLstC that also holds between the counting versions. Here in the counting version of k-NLstC we want to output the number of s-t paths through at most k colors, mod $\lceil 2^{2k\lg^2(n)}\rceil$.

Theorem I.12. *If a $O(|C|^{k-2}|E|^{1-\varepsilon/2})$ or $O(|C|^{k-2-\varepsilon}|E|)$ time algorithm exists for (counting mod $2^{2k\lg^2(n)}$) k-NLstC then a $O(n^{k-\varepsilon})$ algorithm exists for (#)FZkC.*

The conditional lower bound of $(|C|^{k-2}|E|)^{1-o(1)}$ resulting from the above theorem is tight. In Appendix of the full version [DLW20] we give the corresponding algorithm.

k-color Edge Labeled st Connectivity. The k-color Edge Labeled *st* Connectivity problem (k-ELstC) asks for a given acyclic graph with colored edges and given source s and target t, if there is a path from s to t that uses only k colors of edges.

We give conditional hardness for both the decision and counting version of the problem (where the counts are mod a small R). This also implies average-case hardness for the counting mod R problem under all three hardness hypotheses of FGC.

Theorem I.13. *If a $\tilde{O}(|E||C|^{k-1-\varepsilon})$ or $\tilde{O}(|E|^{1-\varepsilon}|C|^{k-1})$ time algorithm exists for (counting mod $2^{2k\lg^2(n)}$) k-*

ELstC, then a $\tilde{O}(n^{k-\varepsilon})$ algorithm exists for (#)Fk-f.

This is tight. Note this algorithm is slower (by a factor of $|C|$) than the node-labeled version, however it is optimal. The corresponding algorithm is in a theorem from the Appendix of the full version [DLW20].

$(k+1)$ Labeled Max Flow. The $(k+1)$ Labeled Max Flow problem studied in [GCSR13] asks, given a capacitated graph $G=(V,E)$ where the edges have colors, and $s,t \in V$, if there is a maximum flow from the source s to the sink t where number of distinct colors of the edges with non-zero flow is at most $k+1$.

Theorem I.14. *If $(k+1)$L-MF can be solved in $T(n)$ time, then we can solve FZkC in time $\tilde{O}(T(n)+n^2)$.*

This implies an $n^{k-1-o(1)}$ lower bound for kL-MF under all three FGC hypotheses. We also show that for the particular structured version of the problem given in our reduction, this lower bound is tight.

Regular Expression Matching. The Regular Expression Matching problem (studied e.g. in [BI16]) takes as input a regular expression (pattern) p of size m and a sequence of symbols (text) t of length n, and asks if there is a substring of t that can be derived from p. The counting version of the problem, #Regular Expression Matching asks for the number of subset alignments of the pattern in the text mod an integer R, where $R=n^{o(1)}$. A classic algorithm constructs and simulates a non-deterministic finite automaton corresponding to the expression, resulting in the rectangular $O(mn)$ running time for the detection version of the problem.

We give hardness from #F2-OV (mod R) which in turn implies average-case fine-grained hardness for counting regular expression matchings mod R, from SETH.

Theorem I.15. *Let R be an integer where $\lg(R)$ is subpolynomial. If you can solve (# mod R) regular expression matching in $T(n)$ time, then you can solve (# mod R) F2-OV in $\tilde{O}(T(n)+n)$ time*

Again, we show in Appendix of the full version [DLW20] that for the particular "type" of pattern used in our reduction, this lower bound is tight.

LCS and Edit Distance. The k-LCS problem is a basic problem in sequence alignment. Given k sequences $s_1,\ldots,s_k$ of length n, one is asked to find the longest sequence that appears in every s_i as a subsequence. k-LCS can be solved in $O(n^k)$ time with dynamic programming and requires $n^{k-o(1)}$ time under SETH, via a reduction from k-OV [ABV15]. Here we show that k-LCS is also fine-grained hard via a reduction from Fk-OV.

Theorem I.16. *A $T(n)$ time algorithm for k-LCS with alphabet size $O(k)$ implies a $\tilde{O}(T(n))$ algorithm for Fk-OV.*

The Edit Distance problem is another famous sequence alignment problem. Here one is given two n length sequences a and b and one needs to compute the minimum number of symbol insertions, deletions and substitutions needed to transform a into b. Edit Distance can be solved in $O(n^2)$ time via dynamic programming, and requires $n^{2-o(1)}$ time under SETH, via a reduction from OV [BI15], [BI18].

In the full version [DLW20] we show that edit distance is also fine-grained hard from F2-OV.

Theorem I.17. *A $T(n)$ time algorithm for Edit Distance implies a $\tilde{O}(T(n))$ algorithm for F2-OV.*

1) Counting OV is Easy on Average: As mentioned earlier in the introduction we show that counting orthogonal vectors over the uniform distribution is easy in the average-case. Let $\#\text{OV}^{\mu,d}$ be the problem of solving orthogonal vectors on instances generated by sampling n vectors iid from the distribution over d bit vectors where every bit in the vector is sampled iid from the distribution that returns 1 with probability μ and returns 0 with probability $1-\mu$.

Theorem I.18. *For all constant values of μ and all values of d there exists constants $\varepsilon>0$ and $\delta>0$ such that there is an algorithm for $\#\text{OV}^{\mu,d}$ that runs in time $\tilde{O}(n^{2-\delta})$ with probability at least $1-n^{-\varepsilon}$.*

2) Counting to Detection for ZkC: Our worst-case to average-case reductions show hardness for counting problems. We mentioned earlier in the introduction that stronger cryptographic primitives have been built from detection problems than from counting problems. In this paper we show that in the sufficiently low error regime there is a counting to detection reduction for the zero-k-clique problem. Unfortunately, this does *not* give a fine-grained one-way function from worst-case assumptions. However, it makes progress towards bridging the gap between the problems we can show hard from the worst-case and those we can build powerful cryptographic primitives from.

Definition 1. An average case instance of ZkC (ACZkC) with range R takes as input a complete k-partite graph with n nodes in each partition. Every edge has a weight chosen iid from $[0,R-1]$. A clique is considered a zero k clique if the sum of the edges is zero mod R.

Theorem I.19. *Given a decision algorithm for ACZkC that runs in time $O(n^{k-\varepsilon})$ for some $\varepsilon > 0$ and succeeds with probability at least $1-n^{-\omega(1)}$, there is a counting algorithm that runs in $O(n^{k-\varepsilon'})$ time for some $\varepsilon' > 0$ and succeeds with probability at least $1-n^{-\omega(1)}$, where $\omega(1)$ here means any function that is asymptotically larger than constant.*

3) Worst-Case to Average-Case Reductions: We define the notion of a good low-degree polynomial for the problem P (a GLDP(P)). We define the properties of a good low-degree polynomial in Definition 8. Intuitively these properties are that the function must be low degree, count the output of the problem, and have well structured monomials. We show that any problem P that has a GLDP(P) is hard in its uniform average case in the full version [DLW20]. We do this using techniques from Boix-Adserà et al [BBB19]. We use the GLDP($\cdot$) framework to show uniform average-case hardness for our counting factored problems (in the full version). We give the framework theorem statement below.

Theorem I.20. *Let μ be a constant such that $0 < \mu < 1$. Let P be a problem such that a function f exists that is a GLDP(P), and let d be the degree of f. Let A be an algorithm that runs in time $T(n)$ such that when $\vec{I}$ is formed by n bits each chosen iid from Ber$[\mu]$:*

$$Pr[A(\vec{I}) = P(\vec{I})] \geq 1 - 1/\omega\left(\lg^d(n)\lg\lg^d(n)\right).$$

Then there is a randomized algorithm B that runs in time $\tilde{O}(n+T(n))$ such that for any *for $\vec{I} \in \{0,1\}^n$:*

$$Pr[B(\vec{I}) = P(\vec{I})] \geq 1 - O\left(2^{-\lg^2(n)}\right).$$

Boix-Adserà et al show that counting k cliques is as hard in Erdős-Rényi graphs as it is in the worst case. We use the GLDP($\cdot$) framework a second time to slightly generalize their result to show that counting any subgraph H in an Erdős-Rényi graph is at least as hard as counting subgraphs H in worst case k-partite graphs (in the full version [DLW20]).

Theorem I.21. *Let H have e edges and k vertices where $k = o(\sqrt{\lg(n)})$. Let A be an average-case algorithm for counting subgraphs H in Erdős-Rényi graphs with edge probability $1/b$ which takes $T(n)$ time with probability $1-2^{-2k}\cdot b^{-k^2}\cdot(\lg(e)\lg\lg(e))^{-\omega(1)}$.*

Then an algorithm exists to count subgraphs H in k-partite graphs in time $\tilde{O}(T(n))$ with probability at least $1-\tilde{O}(2^{-\lg^2(n)})$.

II. Preliminaries

We cover useful preliminaries for the full paper in this section.

A. Hypotheses about Core Problems of Fine-Grained Complexity

Definition 1. The 3-SUM Hypothesis [GO95] In the *k-SUM problem*, we are given an unsorted list L of n values (over $\mathbb{Z}$ or $\mathbb{R}$) and want to determine if there are $a_1,\ldots,a_k \in L$ such that $\sum_{i=1}^{k} a_i = 0$. The counting version of k-SUM asks how many sets of k numbers $a_1,\ldots,a_k \in L$ sum to zero.

The k-SUM hypothesis states that that the k-SUM problem requires $n^{\lceil k/2\rceil - o(1)}$ time [GO95].

This is equivalent to saying no $n^{\lceil k/2\rceil-\varepsilon}$ time algorithm exists for k-SUM for constant $\varepsilon > 0$.

Definition 2. APSP Hypothesis [VW10] APSP takes as input a graph G with n nodes (vertices), V and m edges, E. These edges are given weights in $[-R,R]$ where $R = O(n^c)$ for some constant c. We must return the shortest path length for every pair of vertices $u,v \in V$. The length of a path is the sum of the edge weights for all edges on that path.

The APSP Hypothesis states that the APSP problem requires $n^{3-o(1)}$ time when $m = \Omega(n^2)$.

Definition 3. Strong Exponential Time Hypothesis (SETH) [IP01] Let c_k be the smallest constant such that there is an algorithm for k-CNF SAT that runs in $O(2^{c_k n+o(n)})$ time.

SETH states that there is no constant $\varepsilon > 0$ such that $c_k \leq 1-\varepsilon$ for all constant k.

Intuitively SETH states that there is no constant $\varepsilon > 0$ such that there is a $O(2^{n(1-\varepsilon)})$ time algorithm for k-CNF SAT for all constant values of k.

Definition 4. The k-OV Hypothesis [Wil07] In the *k-OV problem*, we are given k unsorted lists $L_1,\ldots,L_k$ of n zero-one vectors of length d as input. If there are k vectors $v_1 \in L_1,\ldots,v_k \in L_k$ such that for $\forall i \in [1,d]$ $\exists j \in [1,k]$ such that $v_i[j] = 0$ we call these k vectors an orthogonal k-tuple. One should return true if there is an orthogonal k-tuple in the input. The counting version of k-OV (#k-OV) asks for the number of orthogonal k-tuples.

The k-OV hypothesis states that that the k-OV problem requires $n^{k-o(1)}$ time [Wil07].

This is equivalent to saying no $O(n^{k-\varepsilon})$ time algorithm exists for k-OVfor constant $\varepsilon > 0$.

B. Graphs

Definition 5. Let $H = (V_H, E_H)$ be a k-node graph with $V_H = \{x_1,\ldots,x_k\}$.

An H-partite graph is a graph with k partitions $V_1,\ldots,V_k$. This graph must only have edges between nodes $v_i \in V_i$ and $v_j \in V_j$ if e $(x_i,x_j) \in E_H$. (See Figure 2)

C. Good Low-Degree Polynomials

We define the good low-degree polynomial for a problem P ($GLDP(P)$). In the full version [DLW20] we provide a framework which shows that if a problem P has a $GLDP(P)$ then P is hard over the uniform average case. The proof of this framework is a generalization of the proof in Boix et al. [BBB19]. We use this to show average-case hardness for counting versions of factored problems and counting subgraphs in the full version.

Definition 6. Let the polynomial f have n inputs $x_1,\ldots,x_n$. We say f is *strongly d-partite* if one can partition the inputs into d sets $S_1,\ldots,S_d$ such that f can be written as a sum of monomials $\sum_i x_{1,i}\cdots x_{d,i}$, where every variable $x_{j,i}$ is from the partition S_j. That is, if there is a monomial $x_{i_1}^{c_1}\cdots x_{i_k}^{c_k}$ in f then it must be that $c_j = 1$ and for all $j \neq \ell$ if $x_{i_j} \in S_m$ then $x_{i_\ell} \notin S_m$.

Definition 7. Let $P(\vec{I})$ be the correct output for problem P given input $\vec{I}$.

Definition 8. Let n be the input size of the problem P, let P return an integer in the range $[0,p-1]$ where p is a prime and $p < n^c$ for some constant c. A *good low-degree polynomial* for problem P (GLDP(P)) is a polynomial f over a prime finite field F_p where:

- If $\vec{I} = b_1,\ldots,b_n$, then $f(b_1,\ldots,b_n) = f(\vec{I}) = P(\vec{I})$ where b_i maps to either a zero or a one in the prime finite field.
- The function f has degree $d = o(\lg(n)/\lg\lg(n))$.
- The function f is strongly d-partite.

D. Factored Problems

We introduce a more expressive extension of k-SUM, k-OV, k-XOR, and ZkC. At a high level this extension takes every number or vector from the original problems and splits them up into $g = o(\lg(n)/\lg\lg(n))$ groups of numbers or vectors with bit representations of size $b = o(\lg(n))$. If the original numbers had length ℓ, then $\ell \approx b \cdot g$. Then, we allow each group to contain multiple numbers or vectors.

We start by giving a definition of Fk-OV, then we give a small example of F2-OV. Next, we follow up with the analogously defined Fk-SUM ,Fk-XOR, and FZT. Finally, we give algorithms for these problems in the full version [DLW20].

1) Fk-OV, Intuition and Examples: **Definition 9.** A (g,b)-*factored vector* v is defined by g sets $(v[1],\ldots,v[g])$ where each $v[i] \subseteq \{0,1\}^b$ is a set of b-dimensional binary vectors.

For a set of vectors $\vec{w}_1,\ldots,\vec{w}_k$ of the same dimension d, let $isOrthogonalTuple(\vec{w}_1,\ldots,\vec{w}_k)$ return 1 iff $\vec{w}_1,\ldots,\vec{w}_k$ are orthogonal, i.e. iff $\sum_{a=1}^{d}\prod_{j=1}^{k} w_j[a] = 0$, where $w_j[a]$ is the a^{th} bit of the vector w_j.

Now we define a useful operator, $\circ$ for a set $\{Z_1,\ldots,Z_k\}$ where each Z_i is a set of d-dimensional binary vectors as follows.

$$\circ(Z_1,\ldots,Z_k) := \sum_{\vec{w}_1\in Z_1,\ldots,\vec{w}_k\in Z_k} isOrthogonalTuple(\vec{w}_1,\ldots,\vec{w}_k).$$

Now, given k (g,b)-factored vectors $v_1,\ldots,v_k$ the number of orthogonal vectors within those factored vectors is $\circledcirc(v_1,\ldots,v_k) := \Pi_{i=0}^{g-1} \circ (v_1[i],\ldots,v_k[i])$.

The input to Fk-OV is $V_1,\ldots,V_k$, where each V_j is a set of n (g,b)-factored vectors, where $g = o(\lg(n)/\lg\lg(n))$ and $b = o(\lg(n))$. The total number of orthogonal vectors in a given Fk-OV instance is

$$\sum_{v_1,\ldots,v_k\in V_1,\ldots,V_k} \circledcirc(v_1,\ldots,v_k).$$

The Fk-OV problem asks to determine whether $\sum_{v_1,\ldots,v_k\in V_1,\ldots,V_k} \circledcirc(v_1,\ldots,v_k) > 0$.

An Example: We give a small example bellow. Consider F2-OV where $g = 2$ and $b = 3$. We give an example of factored vectors u, v and w:

$$u[0] = \{001,010\} \qquad u[1] = \{001,010\}$$
$$v[0] = \{000,010,110\} \qquad v[1] = \{110,101\}$$
$$w[0] = \{\} \qquad w[1] = \{000,011,100,111\}$$

First, note that $\circledcirc(w,u) = \circledcirc(w,v) = 0$ trivially because $w[0]$ is the empty set. Empty sets are valid in this factored representation, but, rather degenerate. Next, note that $\circledcirc(v,u)$ is $4 \cdot 2 = 8$. For $\circ(u[0],v[0])$ all of $(001,000),(001,010),(001,110)$, and $(010,000)$ are orthogonal. For $\circ(u[1],v[1])$ both $(001,110)$, and $(010,101)$ are orthogonal.

A Natural Interpretation:: We can generate a k$-OV$ instance by interpreting a factored vector as representing $|v_1|\cdot\ldots\cdot|v_k|$ vectors. For example u in the above example would represent the following list of vectors:

$$001001,001010,010001,010010.$$

As another example v would represent the following list of vectors:

$$000110,000101,010110,010101,110110,110101.$$

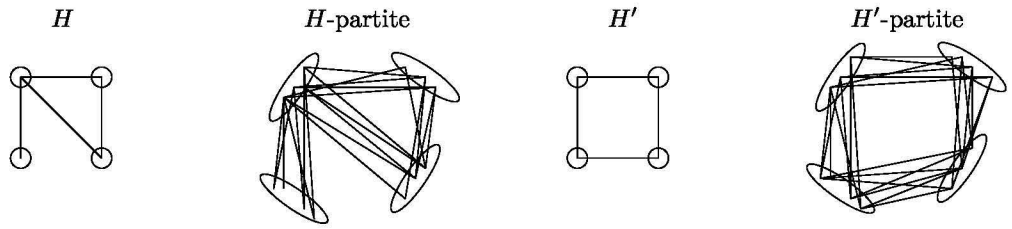

Figure 2: An example of the corresponding H-partite graphs.

Finally, W represents no vectors, because $w[0]$ is the empty set.

However, the number of vectors that can be represented by a single factored vector that has a $g2^b$ sized representation is 2^{bg}. While $g2^b$ is sub-polynomial, 2^{bg} can be super polynomial (e.g. if $b = g = \lg(n)^{3/2}$)!

2) Definitions for Fk-ϝ, Fk-SUM, Fk-XOR, and FZT:

Definition 10. Let $f : (\{0,1\}^b)^{\times k} \to \{0,1\}$ be a function taking k b-dimensional binary vectors to $\{0,1\}$. We can view f as a Boolean function.

Let us define an operator for f, $\circ_f$, that takes k factored vectors $a_1,\ldots,a_k$ and computes the number of k-tuples of vectors, one in each a_i, that f accepts:

$$\circ_f(a_1,\ldots,a_k) = \sum_{\vec{w}_1 \in a_1 \ldots \vec{w}_k \in a_k} f(\vec{w}_1,\ldots,\vec{w}_k).$$

If v is a (g,b)-factored vector let, for $i \in [g]$, $v[i]$ be the i^{th} set of vectors in v.

Given (g,b)-factored vectors $v_1,\ldots,v_k$ the number of k-tuples of vectors accepted by f within those factored vectors is $\circledcirc_f(v_1,\ldots,v_k) = \Pi_{i=0}^{g-1} \circ_f (v_1[i],\ldots,v_k[i])$.

For each f, we define a problem Fk-ϝ. The input to Fk-ϝ is k sets, $V_1,\ldots,V_k$, of n (g,b)-factored vectors each, where $g = o(\lg(n)/\lg\lg(n))$ and $b = o(\lg(n))$.

The total number k-tuples of vectors accepted by f in a given Fk-ϝ instance is

$$\text{Fk-}f(V_1,\ldots,V_k) := \sum_{v_1,\ldots,v_k \in V_1,\ldots,V_k} \circledcirc_f(v_1,\ldots,v_k).$$

The Fk-ϝ problem returns true iff Fk-$f(V_1,\ldots,V_k) > 0$. More generally, the counting version #Fk-ϝ of Fk-ϝ asks to compute the quantity Fk-$f(V_1,\ldots,V_k)$.

Definition 11. Fk-XOR is the problem Fk-ϝ where f is 1 if the componentwise XOR of the k given vectors is the 0 vector:

$$f(v_1,\ldots,v_k) = \begin{cases} 1, & \text{if } v_1 \oplus \ldots \oplus v_k = \vec{0} \\ 0, & \text{else} \end{cases}.$$

Definition 12. Fk-SUM is the problem Fk-ϝ where f that checks if the sum of the k vectors is the 0 vector:

$$f(v_1,\ldots,v_k) = \begin{cases} 1, & \text{if } v_1 + \ldots + v_k = 0 \\ 0, & \text{else} \end{cases}.$$

Definition 13. For an integer k, $\ell = \binom{k}{2}$ and a given function $f : \{0,1\}^{b\ell} \to \{0,1\}$, construed as taking ℓ-tuples of b-length binary vectors to $\{0,1\}$, let #FϝkC be the problem of counting cliques in a graph whose edges are labeled with factored vectors, where a clique is counted with multiplicity the number of ℓ-tuples of vectors that f accepts and that appear in the ℓ factored vectors labeling the edges.

More formally, we change the definition of the operation $\circledcirc_f(\cdot)$ to take as input k vertices $v_1,\ldots,v_k$ of a given graph $G = (V,E)$ whose edges $(x,y) \in E$ are labeled by (g,b)-factored vectors $e_{x,y}$:

$$\circledcirc'_f(v_1,\ldots,v_k) =$$
$$\textit{isClique}(v_1,\ldots,v_k) \cdot \Pi_{i=0}^{g-1} \circ_f (e_{v_1,v_2}[i], e_{v_1,v_3}[i],\ldots,e_{v_{k-1},v_k}[i]).$$

Above $\textit{isClique}(v_1,\ldots,v_k)$ outputs 1 if $v_1,\ldots,v_k$ form a k-clique in G, and otherwise outputs 0.

We keep the definition of $\circ_f(\cdot)$ the same as before, but now its input is a list of ℓ sets of vectors that are the ith group of vectors of the factored vectors labeling the clique edges:

$$\circ_f(e_1[i],\ldots,e_\ell[i]) = \sum_{\vec{w}_1 \in e_1[i] \ \ldots \vec{w}_\ell \in e_\ell[i]} f(\vec{w}_1,\ldots,\vec{w}_\ell).$$

Finally, we let #FϝkCbe the problem of computing

$$\text{F}f k\text{C}(G) := \sum_{v_1,\ldots,v_k \in V} \circledcirc'_f(v_1,\ldots,v_k).$$

Here, unlike for #Fℓ-f, we are only counting the sums of factored vectors when those factored vectors are on a set of $\ell = \binom{k}{2}$ edges that form a k clique. Let FϝkCbe the detection version of the problem that returns 1 if FfkC$(G) > 0$ and 0 otherwise.

Definition 14. Factored Zero k-Clique, FZkC is the F$\mathfrak{f}k$C problem where f is the sum function for $\binom{k}{2}$ variables defined in the definition of Fk-SUM .

Definition 15. Factored Zero Triangle, FZT is FZ3C.

3) Hypotheses for Factored Problems: First we will define the hypotheses for our factored list problems.

In many lemma, theorem and definition statements we will use a structure where we put (#) before several problem or hypothesis names. This structure means that the statement is true for all non counting versions, or for all counting versions. For example, in the first line below the two implies statements are:
"The Fk-OV hypothesis (i.e.Fk-OVH) states that Fk-OV requires $n^{k-o(1)}$ time."
and "The #Fk-OV hypothesis (i.e.#Fk-OVH) states that #Fk-OV requires $n^{k-o(1)}$ time.".

Definition 16. The (#)Fk-OV hypothesis (i.e.(#)Fk-OVH) states that (#) Fk-OV requires $n^{k-o(1)}$ time.

The (#) Fk-SUM hypothesis (i.e.(#)Fk-SUMH) states that (#) Fk-SUM requires $n^{k-o(1)}$ time.

The (#)Fk-XOR hypothesis (i.e.(#)Fk-XORH) states that (#)Fk-XOR requires $n^{k-o(1)}$ time.

The (#)Fk-$\mathfrak{f}$ hypothesis (i.e.(#)Fk-$\mathfrak{f}$H) states that (#)Fk-$\mathfrak{f}$ requires $n^{k-o(1)}$ time.

Now we will define the hypotheses for our factored clique problems.

Definition 17. The (#)FZkC hypothesis (i.e.(#)FZkCH) states that (#) FZkC requires $n^{k-o(1)}$ time.

The (#)F$\mathfrak{f}k$C hypothesis (i.e.(#)F$\mathfrak{f}k$CH) states that (#) F$\mathfrak{f}k$C requires $n^{k-o(1)}$ time.

4) Average-Case for Factored Problems: We will separate the average-case distribution of factored problems into the normal case and a more-general parameterized case.

Definition 18. More General Average-Case Let $S_{b,\mu}$ be a distribution over sets of vectors from $\{0,1\}^b$. A set drawn from $S_{b,\mu}$ includes every vector $w \in \{0,1\}^b$ with probability μ.

Let $D_{g,b,\mu}$ be a distribution over factored vectors v where all g sets of $v[i]$ are sampled iid from $S_{b,\mu}$.

The average-case distribution for #Fk-$\mathfrak{f}^{\mu}$ samples every factored vector in its input iid from $D_{g,b,\mu}$.

The average-case distribution for #F$\mathfrak{f}k$C$^{\mu}$ samples every factored vector in its input iid from $D_{g,b,\mu}$.

For the average-case we use in this paper we use $\mu = 1/2$. We feel this is the most natural distribution for our problem. We will occasionally call this the "uniform average-case" to emphasize that every set $v[i]$ in every factored vector is chosen uniformly at random from all possible subsets of $\{0,1\}^b$.

Definition 19. The average-case distribution for #Fk-$\mathfrak{f}$ samples every factored vector in its input iid from $D_{g,b,1/2}$.

The average-case distribution for #F$\mathfrak{f}k$C samples every factored vector in its input iid from $D_{g,b,1/2}$.

ACKNOWLEDGEMENTS

We would like to acknowledge Marshall Ball for interesting and helpful early discussions.

We would like to acknowledge all our reviewers for helpful comments. We thank all the reviewers for advice about improving the readability of the paper. We would like to extend special thanks to reviewer 1 who noted that we could improve the definition of good low-degree polynomial by removing a restriction!

REFERENCES

[ABV15] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Tight hardness results for LCS and other sequence similarity measures. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 59–78, 2015.

[AVY18] Amir Abboud, Virginia Vassilevska Williams, and Huacheng Yu. Matching triangles and basing hardness on an extremely popular conjecture. *SIAM Journal on Computing*, 47(3):1098–1122, 2018.

[BBB19] Enric Boix-Adserà, Matthew Brennan, and Guy Bresler. The average-case complexity of counting cliques in erdős-rényi hypergraphs. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 1256–1280. IEEE Computer Society, 2019.

[BI15] Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 51–58. ACM, 2015.

[BI16] Arturs Backurs and Piotr Indyk. Which regular expression patterns are hard to match? In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 457–466. IEEE, 2016.

[BI18] Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). *SIAM J. Comput.*, 47(3):1087–1097, 2018.

[BRSV17] Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. Average-case fine-grained hardness. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 483–496. ACM, 2017.

[BRSV18] Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. Proofs of work from worst-case assumptions. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 789–819. Springer, 2018.

[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 Madhu Sudan, editor, *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, pages 261–270, 2016.

[DLW20] Mina Dalirrooyfard, Andrea Lincoln, and Virginia Vassilevska Williams. New techniques for proving fine-grained average-case hardness. *CoRR*, abs/2008.06591, 2020.

[GCSR13] Donatella Granata, Raffaele Cerulli, Maria Grazia Scutellà, and Andrea Raiconi. Maximum flow problems and an np-complete variant on edge-labeled graphs. *Handbook of Combinatorial Optimization*, pages 1913–1948, 2013.

[GO95] Anka Gajentaan and Mark H. Overmars. On a class of o(n2) problems in computational geometry. *Comput. Geom.*, 5:165–185, 1995.

[GR18] Oded Goldreich and Guy N. Rothblum. Counting t-cliques: Worst-case to average-case reductions and direct interactive proof systems. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 77–88. IEEE Computer Society, 2018.

[GR20] Oded Goldreich and Guy N. Rothblum. Worst-case to average-case reductions for subclasses of P. In *Computational Complexity and Property Testing - On the Interplay Between Randomness and Computation*, volume 12050 of *Lecture Notes in Computer Science*, pages 249–295. Springer, 2020.

[IP01] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001.

[KW19] Daniel M. Kane and R. Ryan Williams. The orthogonal vectors conjecture for branching programs and formulas. In Avrim Blum, editor, *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, volume 124 of *LIPIcs*, pages 48:1–48:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2019.

[LLV19] Rio LaVigne, Andrea Lincoln, and Virginia Vassilevska Williams. Public-key cryptography in the fine-grained setting. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 605–635. Springer, 2019.

[Vas18] Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the ICM*, volume 3, pages 3431–3472. World Scientific, 2018.

[VW10] Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 645–654. IEEE Computer Society, 2010.

[VW13] Virginia Vassilevska Williams and Ryan Williams. Finding, minimizing, and counting weighted subgraphs. *SIAM J. Comput.*, 42(3):831–854, 2013.

[VW18] Virginia Vassilevska Williams and R. Ryan Williams. Subcubic equivalences between path, matrix, and triangle problems. *J. ACM*, 65(5):27:1–27:38, 2018.

[Wil07] Ryan Williams. Algorithms and resource requirements for fundamental problems. *Ph. D. dissertation, Ph. D. Thesis*, 2007.