2018 IEEE 59th Annual Symposium on Foundations of Computer Science

Counting t-Cliques: Worst-Case to Average-Case Reductions and Direct Interactive Proof Systems

Oded Goldreich
Department of Computer Science
Weizmann Institute of Science
Rehovot, Israel
oded.goldreich@weizmann.ac.il

Guy N. Rothblum
Department of Computer Science
Weizmann Institute of Science
Rehovot, Israel
rothblum@alum.mit.edu

***Abstract*—We study two aspects of the complexity of counting the number of t-cliques in a graph:**

1) ***Worst-case to average-case reductions*: Our main result reduces counting t-cliques in any n-vertex graph to counting t-cliques in typical n-vertex graphs that are drawn from a simple distribution of min-entropy $\widetilde{\Omega}(n^2)$. For any constant t, the reduction runs in $\widetilde{O}(n^2)$-time, and yields a correct answer (w.h.p.) even when the "average-case solver" only succeeds with probability $1/\mathrm{poly}(\log n)$.**
2) ***Direct interactive proof systems*: We present a direct and simple interactive proof system for counting t-cliques in n-vertex graphs. The proof system uses $t-2$ rounds, the verifier runs in $\widetilde{O}(t^2n^2)$-time, and the prover can be implemented in $\widetilde{O}(t^{O(1)} \cdot n^2)$-time when given oracle access to counting $(t-1)$-cliques in $\widetilde{O}(t^{O(1)} \cdot n)$-vertex graphs.**

The results are both obtained by considering weighted versions of the t-clique problem, where weights are assigned to vertices and/or to edges, and the weight of cliques is defined as the product of the corresponding weights. These weighted problems are shown to be easily reducible to the unweighted problem.

***Keywords*-counting cliques; worst-case to average-case reductions; interactive proofs, fine-grained complexity;**

I. Introduction

We study two (seemingly unrelated) aspects of the complexity of counting t-cliques. The first study refers to the relation between the worst-case complexity of problems in $\mathcal{P}$ and their average-case complexity. The second study seeks (direct and intuitive) doubly-efficient interactive proof system for problems in $\mathcal{P}$. Indeed, both studies are related to the recently emerging theory of "hardness within $\mathcal{P}$" [31], and counting t-cliques in graphs is a good test case for both studies for several reasons:

1) Counting t-cliques in (n-vertex) graphs is a natural candidate for "hardness within $\mathcal{P}$" (i.e., it is in $\mathcal{P}$ and is assumed to have worst-case complexity $n^{\Theta(t)}$);
2) Counting t-cliques is a well-studied and natural problem; and
3) Counting t-cliques has an appealing combinatorial structure (which is indeed capitalized upon in our work).

In Sections I-A and I-B we discuss each study seperately, whereas Section I-C reveals the common themes that lead us to present these two studies in one paper. We direct the reader to the full version of this work [18] for full details.

A. Worst-case to average-case reductions

1) Background: While most research in the theory of computation refers to worst-case complexity, the importance of average-case complexity is widely recognized (cf., e.g., [13, Chap. 1–10.1] versus [13, Sec. 10.2]). Worst-case to average-case reductions, which allow for bridging the gap between the two theories, are of natural appeal (to say the least). Unfortunately, worst-case to average-case reductions are known only either for "very high" complexity classes, such as $\mathcal{E}$ and $\#\mathcal{P}$ (see [4] and [23], [10], [15][1], resp.), or for "very low" complexity classes, such as $\mathcal{AC}_0$ (cf. [2], [3]). In contrast, presenting a worst-case to average-case reduction for $\mathcal{NP}$ is a well-known open problem, which faces significant obstacles as articulated in [12], [9].

In the context of fine-grained complexity. A recent work by Ball, Rosen, Sabin, and Vasudevan [5] initiated the study of worst-case to average-case reductions in the context of fine-grained complexity.[2] The latter context focuses on the exact complexity of problems in $\mathcal{P}$ (see, e.g., the survey by V. Williams [31]), attempting to classify problems into classes of similar polynomial-time complexity (and distinguishing, say, linear-time from quadratic-time and cubic-time). Needless to say, reductions used in the context of

[1]The basic idea underlying the worst-case to average-case reduction of the "permanent" is due to Lipton [23], but his proof implicitly presumes that the field is somehow fixed as a function of the dimension. This issue was addressed independently by [10] and in the proceeding version of [15]. In the current work, we shall be faced with the very same issue.

[2]In retrospect, as will be discussed shortly below, some prior results can be reinterpreted as belonging to this setting.

2575-8454/18/$31.00 ©2018 IEEE
DOI 10.1109/FOCS.2018.00017

fine-grained complexity must preserve the foregoing classification, and the simplest choice – taken in [5] – is to use almost linear-time reductions.

The pioneering paper of Ball *et al.* [5] shows that there exist (almost linear-time) reductions from the worst-case of several natural problems in $\mathcal{P}$, which are widely believed to be "somewhat hard" (i.e., have super-linear worst-case time complexity), to the average-case of some other problems that are in $\mathcal{P}$. In particular, this is shown for the Orthogonal Vector problem, for the 3-SUM problem, and for the All Pairs Shortest Path problem. Hence, the worst-case complexity of problems that are widely believed to be "somewhat hard" is reduced to the average-case complexity of problems in $\mathcal{P}$. Furthermore, the worst-case complexity of the latter problems matches (approximately) the best algorithms known for the former problems (although the actual worst-case complexity of these problems may in fact be lower).

In our prior work [17], we tighten the foregoing result by defining, for each polynomial p, a worst-case complexity class $\mathcal{C}^{(p)}$ that is a subset of $\mathrm{Dtime}(p^{1+o(1)})$, and showing, for any problem Π in $\mathcal{C}^{(p)}$, an almost linear-time reduction from solving Π in the *worst-case* to solving a different problem $\Pi' \in \mathcal{C}^{(p)}$ in the *average-case*. Furthermore, we showed that $\mathcal{C}^{(p)}$ contains problems whose average-case complexity almost equals their worst-case complexity.

Loosely speaking, the class $\mathcal{C}^{(p)}$ consists of counting problems that refer to $p(n)$ local conditions regarding the n-bit long input, where each local condition refers to $n^{o(1)}$ bit locations and can be evaluated in $n^{o(1)}$-time. In particular, for any constant $t > 2$ and $p_t(n) = n^t$, the class $\mathcal{C}^{(p_t)}$ contains problems such as t-CLIQUE and t-SUM.

We emphasize that the foregoing results present worst-case to average-case reductions for *classes of problems* (i.e., reducing the worst-case complexity of one problem to the average-case complexity of another problem). Hence, the foregoing results leave open the question whether there exists an almost linear-time worst-case to average-case reduction for a *problem* in $\mathcal{P}$, let alone a natural problem of conjectured high worst-case complexity.

Worst-case to average-case reductions for individual problems. As stated in Footnote 2, some prior results can be reinterpreted as worst-case to average-case reductions for seemingly hard problem in $\mathcal{P}$. In particular, this holds for problems shown to be random self-reducible. An archetypical case, presented by Blum, Luby, and Rubinfeld [8], is that of matrix multiplication: the product AB is computed as $(A+R)(B+S)-(A+R)S-R(B+S)+RS$, where R and S are random matrices. Note, however, that in this case the potential hardness of the problem is quite modest (as $n \times n$ matrix products can be computed in time $o(n^{2.373})$).

A far less known case was presented by Goldreich and Wigderson [19], who considered the problem of computing the function $f_n(A_1, ..., A_{\ell(n)}) = \sum_{S \subseteq [\ell(n)]} \mathtt{DET}(\sum_{i \in S} A_i)$, where the A_i's are n-by-n matrices over a finite field and $\ell(n) = O(\log n)$. They conjectured that this function cannot be computed in time $2^{\ell(n)/3}$, and showed that it is random self-reducible (by $O(n)$ queries).[3] Note, however, that the foregoing problem is not as well-studied as the problems considered in [5], [17], including counting t-cliques.

2) Our results: Our main result in this work is reducing the worst-case complexity of counting t-cliques to the average-case complexity of counting t-cliques. Doing so, we show that the worst-case and average-case complexity of counting t-cliques are essentially equal.

Theorem 1.1: (worst-case to average-case reduction for counting cliques of given size): For any constant t, there exists a simple distribution on n-vertex graphs and a $\widetilde{O}(n^2)$-time worst-case to average-case reduction of counting t-cliques in n-vertex graphs to counting t-cliques in graphs generated according to this distribution such that the reduction outputs the correct value with probability $2/3$ provided that the error rate *(of the average-case solver)* is a constant smaller than one fourth. Furthermore, the reduction makes $\mathrm{poly}(\log n)$ queries, and the distribution $\mathcal{G}_n$ can be generated in $\widetilde{O}(n^2)$-time and is uniform on a set of $\exp(\widetilde{\Omega}(n^2))$ graphs.

We obtain a similar reduction for counting (simple) t-cycles (for odd $t \geq 3$), see the full version [18] for details.

The notion of average-case complexity that underlies the foregoing discussion (and Theorem 1.1) refers to solving the problem on at least a 0.76 fraction of the instances. This notion may also be called *typical-case complexity*. A much more relaxed notion, called *rare-case complexity*, refers to solving the problem on a noticeable[4] fraction of the instances (say, on a $1/poly(\log n)$ fraction of the n-bit long instances).

Theorem 1.2: (worst-case to rare-case reduction for counting cliques): For any constant t, there exists a simple distribution on n-vertex graphs and a $\widetilde{O}(n^2)$-time worst-case to rare-case reduction of counting t-cliques in n-vertex graphs to counting t-cliques in graphs generated according to this distribution such that the reduction outputs the correct value with probability $2/3$ provided that the success rate *(of the rare-case solver)* is at least $1/\mathrm{poly}(\log n)$. Furthermore, the reduction makes $\widetilde{O}(n)$ queries, and the distribution $\mathcal{G}_n$ can be generated in $\widetilde{O}(n^2)$-time and is uniform on a set of $\exp(\widetilde{\Omega}(n^2))$ graphs.

Theorem 1.2 is meaningful only for $t > 3$ (since the reduction makes $\Omega(n)$ queries regarding (related) instances

[3] They also showed that it is downwards self-reducible when the field has the form $GF(2^{m(n)})$ such that $m(n) = 2^{\lceil \log_2 n \rceil}$ (or $m(n) = 2 \cdot 3^{\lceil \log_3 n \rceil}$).

[4] Here a "noticeable fraction" is the ratio of a linear function over an almost linear function. We stress that this is not the standard definition of this notion (at least not in cryptography).

of description length $\Omega(n^2)$).[5]

B. Doubly-efficient interactive proof systems

1) Background: The notion of interactive proof systems, put forward by Goldwasser, Micali, and Rackoff [21], and the demonstration of their power by Lund, Fortnow, Karloff, and Nisan [24] and Shamir [27] are among the most celebrated achievements of complexity theory. Recall that an interactive proof system for a set S is associated with an interactive verification procedure, V, that can be made to accept any input in S but no input outside of S. That is, there exists an interactive strategy for the prover that makes V accepts any input in S, but no strategy can make V accept an input outside of S, except with negligible probability. (See [13, Chap. 9] for a formal definition as well as a wider perspective.)

The original definition does not restrict the complexity of the strategy of the prescribed prover and the constructions of [24], [27] use prover strategies of high complexity. This fact limits the applicability of these proof systems in practice. (Nevertheless, such proof systems may be actually applied when the prover knows something that the verifier does not know, such as an NP-witness to an NP-claim, and when the proof system offers an advantage such as zero-knowledge [21], [14].)

Doubly-efficient proof systems. Seeking to make interactive proof systems available for a wider range of applications, Goldwasser, Kalai and Rothblum put forward a notion of *doubly-efficient* interactive proof systems (also called *interactive proofs for muggles* [20] and *interactive proofs for delegating computation* [26]). In these proof systems the prescribed prover strategy can be implemented in polynomial-time and the verifier's strategy can be implemented in almost-linear-time. (We stress that unlike in *argument systems*, the soundness condition holds for all possible cheating strategies, and not only for feasible ones.) Restricting the prescribed prover to run in polynomial-time implies that such systems may exist only for sets in $\mathcal{BPP}$, and thus a polynomial-time verifier can check membership in such sets by itself. However, restricting the verifier to run in almost-linear-time implies that something can be gained by interacting with a more powerful prover, even though the latter is restricted to polynomial-time.

The potential applicability of doubly-efficient interactive proof systems was demonstrated by Goldwasser, Kalai and Rothblum [20], who constructed such proof systems for any set that has log-space uniform circuits of bounded depth (e.g., log-space uniform $\mathcal{NC}$). A recent work of Reingold, Rothblum, and Rothblum [26] provided such (constant-round) proof systems for any set that can be decided in polynomial-time and a bounded amount of space (e.g., for all sets in $\mathcal{SC}$).

Towards algorithmic design of proof systems. In our prior work [16], we proposed to develop a more "algorithmic" understanding of doubly-efficient interactive proofs; that is, to identify structures and patterns that facilitate the design of efficient proof systems. Specifically, we identified a natural class of polynomial-time computations, and constructed simpler doubly-efficient proof systems for this class.[6] The aforementioned class consists of all sets that can be locally-characterized by the conjunction of polynomially many local conditions, each of which can be expressed by Boolean formulae of polylogarithmic size. The class of locally-characterizable sets is believed not to be in $\mathrm{Dtime}(p)$ for any fixed polynomial p, and contains natural problems of interest such as determining whether a given graph *does not* contain a clique of constant size t.

The proof system presented in [16] capitalizes on the fact that membership in a locally characterizable set can be cast as satisfying polynomially many low-degree equations. Hence, analogously to [24], [27], the first step is recasting membership in a locally-characterizable set as an algebraic problem, which consists of computing the sum of polynomially many evaluations of a low-degree polynomial (where the particular polynomial is derived from the description of the locally-characterizable set). The interactive proof uses the sum-check protocol [24] to verify the correctness of the sum, and the same applies to counting versions of locally charcatizeable sets. Hence, the intuitive appeal of the problem of counting t-cliques is lost at the first step (in which we consider an algebraic version or extension of the original problem).

2) Our results: In the current work, we present a new (doubly-efficient) interactive proof for counting t-cliques in a graph. The proof system proceeds in iterations, where in the i^{th} iteration verifying *the number of* $(t-i+1)$*-cliques in a graph* is reduced to *verifying the number of* $(t-i)$*-cliques in a related graph*. Hence, the claims made in these iterations, whose complexity gradually decreases from one iteration to the next, all have a clear intuitive meaning that is similar to the original problem (counting the number of cliques in a graph).

Beyond its conceptual appeal, this proof system has the concrete advantage that the (honest) prover's complexity is directly related to the complexity of the statement being proved: the prover can be implemented in linear time given an oracle for counting t-cliques. The proof system extends to varying $t = t(n)$, yielding an alternative interactive proof

[5] The time bound of the reduction itself refers to a model of computation in which the cost of making the (equal length) queries $q_1, ..., q_m$ is $|q_1| + \sum_{j\in[n-1]} \Delta(q_j, q_{j+1})$, where $\Delta(x, y)$ denotes the Hamming distance between x and y. This model is justified by the actual cost of composing the reduction with a procedure that solves the reduced instances.

[6] Indeed, the aforementioned class (of locally-characterizable sets) is a sub-class of $\mathcal{NC} \cap \mathcal{SC}$, yet the interactive proofs presented in [16] are significantly simpler than those in [20], [26].

system for $\#\mathcal{P}$; in particular, we note that this proof system does not use the sum-check protocol of [24].

Theorem 1.3: (interactive proof systems for counting cliques of given size): For an efficiently computable function $t : \mathbb{N} \to \mathbb{N}$, let S_t denote the set of pairs (G, N) such that the graph $G = ([n], E)$ has N distinct $t(n)$-cliques. Then, S_t has a $(t-2)$-round (public coin) interactive proof system (of perfect completeness) in which the verifier's running time is $\widetilde{O}(t(n)^2 \cdot n^2)$, and the prover's running time is $\widetilde{O}(t(n)^2 \cdot n^{1+\omega_{\mathtt{mm}} \cdot \lceil (t(n)-1)/3 \rceil})$, where $\omega_{\mathtt{mm}}$ is the matrix multiplication exponent. Furthermore, the prover can be implemented in $\mathrm{poly}(t(n)) \cdot \widetilde{O}(n^2)$-time, when given access to an oracle for counting $(t(n)-1)$-cliques in $\mathrm{poly}(t(n)) \cdot \widetilde{O}(n)$-vertex graphs.

High-level structure of the new interactive proof system. As discussed above, the interactive proof proceeds in iterations. The i^{th} iteration is a (doubly efficient and interactive) reduction from verifying *the number of* $(t-i+1)$*-cliques in a graph* to *verifying the number of* $(t-i)$*-cliques in a related graph*. This proceeds in two conceptual steps.

First, we reduce verifying the number of t'-cliques in $G' = ([n'], E')$ to verifying, for each vertex $j \in [n']$, the number of t'-cliques in G' that contain the vertex j, which equals the number of $(t'-1)$-cliques in the graph induced by the neighbors of j. Next, we let the parties reduce the latter n' claims to a single claim regarding the number of $(t'-1)$-cliques in a new graph, which has the same number of vertices as G', where the reduction is via a single-round randomized interaction. That is, while the first step employs downwards reducibility, where the decreased parameter is the size of the counted cliques, the second step employes *batch verification* (cf., [26]), where *the verification of* n' *claims is reduced to a verification of a single claim of the same type*.

Essentially, the foregoing batch verification is performed by considering an error correcting encoding of the n' graphs by a sequence of $\mathrm{poly}(n')$ graphs, each having n' vertices. The prover sends a succinct description of the number of $(t'-1)$-cliques in these $\mathrm{poly}(n')$ graphs, and the verifier verifies that the sum of the values associated with the n' former graphs equals the claim regarding the number of t-cliques in G'. If so, then the verifier select one of the $\mathrm{poly}(n')$ graphs at random for the next iteration (in which it verifiers the number of $(t'-1)$-cliques in the selected graph). Indeed, the crucial point is finding a suitable encoding scheme, and this is discussed in Section I-C1.

We stress that the foregoing procedure does not use the sum-check protocol, and that each iteration starts and ends with an intuitive combinatorial claim to be verified. (Algebra "raises its ugly head" only in the encoding scheme, which utilizes a so-called multiplication code [25], and in the fact that cliques are indicated by products of all corresponding "edge indicators".) As noted above, a concrete advantage of the current interactive proof system over the one in [16] is that the prover's complexity is proportional to the complexity of counting t-cliques (which is lower than $n^{2.373 \cdot \lceil t/3 \rceil}$) rather than being proportional to n^t. The foregoing procedure works also for varying $t = t(n)$, yielding an alternative interactive proof system for $\#\mathcal{P}$. (We comment that the resulting interactive proof system bears some similarity to the original interactive proof system for the permanent presented in [24]; see further discussion in the full version [18].)

Detour: On a scaled-down version of the permanent. We mention that, under the exponential-time hypothesis, a scaled-down version of the permanent, in which we consider computing the permanent of an ℓ-by-ℓ matrix padded to length $n = 2^{\ell/O(1)}$, constitutes a relatively hard problem in $\mathcal{P}$. Furthermore, the original interactive proof system for the permanent can be adapted to yield a doubly-efficient interactive proof system with $O(\log \ell)$ rounds. An alternative proof system that uses a constant number of rounds is presented in the full version.

Application to proofs of work. Proofs of work were introduced by Dwork and Naor [11] as a method for certifying, in an easily verifiable way, that an untrusted party expended non-trivial computational resources. This may be useful, for example, in fighting denial of service attacks. A proof of work usually consists of a procedure for generating puzzles that are moderately hard to solve, and a doubly-efficient procedure for verifying the correctness of solutions. Our results immediately yield proofs of work based on the problem of counting t-cliques. Puzzles can be generated by sampling graphs from the hard distribution of Theorem 1.1, where the solution is the number of t-cliques in the graph. Solutions can be verified using the interactive proof system of Theorem 1.3. In particular, this yields an appealing proof of *useful* work system (cf. [6]). Moreover, our results imply that the work needed to solve puzzles and to prove the correctness of solutions is closely related to the complexity of counting t-cliques in worst-case graphs. Another desired feature of proofs of work is the non-existence of efficient batching algorithms (a.k.a. "non-amortization" [6]). Our techniques provide a result that partially addresses this concern (see the full version [18] for details).

C. Techniques

One common theme in the two parts of this work is that we find it beneficial to consider "weighted generalizations" of the problem of counting t-cliques. Specifically, we consider graphs with either vertex or edge weights, and define the weight of the clique as the *product* of the corresponding weights (where arithmetic is performed over a finite field). Our definition stands in contrast to the standard practice of defining the weight of a set as the sum of its elements

(cf. [1]), but in the case of vertex-weights it has a very appealing interpretation (see Section I-C1). In any case, after working with the weighted problems, we present reductions from the weighted problems back to the original problem (of counting t-cliques). The reduction for the case of weighted edges is more complex than the one for weighted vertices.

A second common theme is the manipulation of graphs via the manipulation of their vertex (or edge) weights. Encoding a sequence of weighted graphs is performed by encoding the corresponding sequences of weights (see Section I-C1), and performing self-correction is possible by considering sequences that extend each of the weights (see Section I-C2). These comments will hopefully become more clear when we get to the specifics.

1) For the direct interactive proof systems: Here it is useful to consider vertex-weighted graphs. A n-vertex graph with vertex weights $(w_1, ..., w_n)$ such that vertex $i \in [n]$ is assigned the weight w_i may be thought of as a succinct representation of a larger graph consisting of n independent sets such that the i^{th} independent set has w_i vertices and the edges represent complete bipartite graphs between the corresponding independent sets. From this perspective, it is natural to define the weight of the clique S as $\prod_{i \in S} w_i$.

Given such a weighted graph $G = ([n], E)$, the set of sum of the weights of the t-cliques that contain a specific vertex $j \in [n]$ equals w_j times the sum of the weighted $(t-1)$-cliques in the graph induced by the neighbors of j. The latter graph is obtained by resetting the weights of non-neighbors of j to 0 and keeping the weights of the neighbors of j intact. Note that the topology of the graph (i.e., its vertex and edge sets) remain intact, only the weights are updated; that is, the weights $w = (w_1, ..., w_n)$ are replaced by the weights $w^{(j)} = (w_1^{(j)}, ..., w_n^{(j)})$ such that $w_i^{(j)} = w_i$ if $\{i, j\} \in E$ and $w_i^{(j)} = 0$ otherwise. Next, we encode the resulting sequence of weighted graphs, all having the same topology G, by encoding the n weights of each vertex such that the k^{th} codeword, denoted $\overline{c}_k$, encodes the weights of $k \in [n]$ in each of the n graphs (i.e., $w_k^{(1)}, ..., w_k^{(n)}$). Specifically, using the Reed-Solomon code (which is a "good" linear "multiplication code" [25]), we obtain n codewords such that multiplying any t of them (in a coordinatewise manner) yields an (m-long) encoding of the weights of the corresponding t-subset in the n graphs.[7]

In the corresponding iteration of the interactive proof system, the prover sends a codeword, denoted $c = (c_1, ..., c_m)$, that represents the sum of the $\binom{n}{t}$ codewords that encode the weights of all t-subsets of $[n]$. The verifier will decode this (m-long) codeword, check that the sum of the resulting n values equals the value claimed in this iteration, and send the prover a random position in the codeword (i.e., a random $r \in [m]$). The next iteration will refer to the weighted graph G with weights that are determined by the r^{th} coordinate of the codewords $\overline{c}_1, ..., \overline{c}_n$ (i.e., the weights are $\overline{c}_1[r], ..., \overline{c}_n[r]$), and the claimed value will be c_r (i.e., the r^{th} coordinate of the codeword sent by the prover).

Note that we capitalize on the fact that the sum of the weights of t-cliques in a weighted graph is expressed as a sum of t-way products of vertex weights. A crucial feature of the Reed-Solomon code, which enables the foregoing manipulation, is that multiplying together t codewords of the original code that has large distance (i.e., $1 - \epsilon$) yield a codeword of a code that has sufficiently large distance (i.e., $1 - t\epsilon$). And we also use the fact that the code is linear, but *c'est tout*. In particular, unlike Meir [25], we do not use tensor codes. Furthermore, unlike most work in the area, we do not use the sum-check protocol nor refer to objects (like low-degree extensions of Boolean functions) that have no direct intuitive meaning.

The foregoing description refers to vertex weights that reside in a finite field of polynomial (in n) size. To get back to the unweighted problem of counting t-cliques, we first reduce the weighted problem over $\mathrm{GF}(p)$ to $O(t^2 \log p)$ weighted problems over smaller fields, each of size $O(t^2 \log p)$. Next, we reduce each of these problems to an unweighted problem using the reduction outlined in the first paragraph of this section (i.e., replacing each vertex by an independent set of size that equals the vertex's weight and connecting vertices that reside in different independent sets if and only if the original vertices were connected). Since the weights are currently small, this blows-up the size of the graph by a small amount.

2) For the worst-case to average-case reductions: Here it is useful to considered edge-weighted graphs. In fact, we may ignore the graph and just consider weights assigned to all edges of the complete graph, since the non-existence of an edge can be represented by a weight of zero. Hence, we consider a symmetric matrix $W = (w_{j,k})$, and allow non-zero diagonal entries as representations of vertex-weights (as in Section I-C1).[8] We define the weight of the set of vertices S as a product of the weights of all edges that are incident at S (i.e., $\prod_{j \leq k \in S} w_{j,k}$). Similarly to Section I-C1, we show that the sum of the weights of all t-subsets of $[n]$ is proportional to the number of t-cliques in a (much) larger graph, but the reduction in this case is more complex than in Section I-C1. Before discussing this reduction, we outline the ideas used in the worst-case to average-case and rare-case reductions of the edge-weighted problems.

The worst-case to average-case reduction. We first observe that the problem of computing the sum of the weights of t-subsets of vertices in an edge-weighted graph, when defined

[7] Note that if all original weights are in $\{0, 1, ..., b\}$, then we can work with a prime field of size $p = O(n^t b^t)$, since the weight of each t-subset resides in $[0, b^t]$. In subsequent iterations, the claims will refer to the value modulo p. Recall that in this case $m = p$.

[8] Alternatively, one may consider the weights of the diagonal entries as weights of the corresponding self-loops.

over a finite field, is random self-reducible. Specifically, given a n-by-n matrix $W = (w_{j,k})$ such that all $w_{j,k}$'s reside in $\{0, 1, ..., b\}$, we pick a prime field of size $p = O(n^t b^{t^2})$, select uniformly a random matrix $R \in \mathrm{GF}(p)^{n \times n}$, and obtain the values of the sum of weighted t-subsets for the matrices $W + iR$, for $i = 1, ..., t^2$. Note that we are interested in $\mathtt{val}(W)$, where $\mathtt{val}(X)$ is the sum over all t-subset S of $\prod_{j \leq k \in S} x_{j,k}$ (reduced modulo p). Hence, $\mathtt{val}(W + \zeta R)$ is a polynomial of degree $\binom{t}{2} + t < t^2$ in ζ, and its value at 0 can be determined based on its value at $1, ..., t^2$. Furthermore, for every $i \in \mathrm{GF}(p) \setminus \{0\}$, the matrix $W + iR$ is uniformly distributed in $\mathrm{GF}(p)^{n \times n}$. Using $O(t^2)$ non-zero evaluation points (for this polynomial) and employing the Berlekamp–Welch algorithm, we obtain a worst-case to average-case reduction for computing $\mathtt{val}$ modulo p.

The foregoing presentation refers to a fixed prime $p > n^t$, whereas it is not known how to determine such a prime in time $\widetilde{O}(n^2)$. Instead, we pick primes of the desired size at random, apply the self-correction process in each of the corresponding fields, and combine the results using Chinese Remaindering with error [15]. for details, see Section II-B2.

The worst-case to rare-case reduction. Turning from average-case to rare-case targeted reductions, we employ a methodology heralded by Impagliazzo and Wigderson [22], and stated explicitly in our prior work [17]. The methodology is pivoted at the notion of sample-aided reductions, which is extended in the current work. In the following definition (which is taken from [17]), a task consists of a computational problem along with a required performance guarantee (e.g., "solving problem Π on the worst-case" or "solving Π with success rate ρ under the distribution $\mathcal{D}$"). For sake of simplicity, we consider the case that the first task is a worst-case task.

Definition 1.4: (sample-aided reductions): Let $\ell, s : \mathbb{N} \to \mathbb{N}$, and suppose that M is an oracle machine that, on input $x \in \{0,1\}^n$, obtains as an auxiliary input a sequence of $s = s(n)$ pairs of the form $(r, v) \in \{0,1\}^{n+\ell(n)}$. We say that M is a sample-aided reduction of solving Π in the worst-case to the task $\mathtt{T}$ if, for every procedure P that performs the task $\mathtt{T}$, it holds that with probability at least 2/3 over the choice of $r_1, ..., r_s$ it is the case that, for every input $x \in \{0,1\}^n$:

$$\Pr[M^P(x; (r_1, \Pi(r_1)), ..., (r_s, \Pi(r_s))) = \Pi(x)] \geq 2/3,$$

where this "internal" probability is taken over the coin tosses of the machine M and the procedure P. Note that a sample-aided reduction implies an ordinary non-uniform reduction. Furthermore, coupled with a suitable downwards self-reduction for Π, a sample-aided reduction of solving Π in the worst-case to solving Π on the average (resp., in the rare-case) implies a corresponding standard reduction (of worst-case to average-case (resp., to rare-case)).

In this work we extend Definition 1.4 by allowing the sample to be drawn from an arbitrary distribution over $(\{0,1\}^n)^{s(n)}$; in particular, the $s(n)$ individual samples need not be independently and identically distributed. We note that both the foregoing implications hold also under this extension, where for the second implication we also require that the sample distribution be efficiently sampleable.

Our worst-case to rare-case reduction utilizes the worst-case to rare-case reduction of Sudan, Trevisan, and Vadhan [29], which applies to low-degree polynomials. We first observe that this reduction yields a sample-aided reduction, and then apply it to the edge-weighted clique-counting problem, while presenting a downwards self-reduction for the latter problem (where this reduction is analogous to the one used in Section I-C1). We stress that our sample-aided reduction utilizes correlated random samples (and the extension of Definition 1.4 is used here); specifically, in our application, the samples correspond to pairs of objects and we need multiple samples that coincide on the first element of the pair, for multiple choices of such first element.

Back to the unweighted problem. Having established the (average-case and) rare-case hardness of the edge-weighted clique-counting problem, we seek to establish this result for the original counting problem (which refers to simple unweighted graphs). An adequate reduction is presented in Section II-A; it is more complex than the reduction employed to the vertex-weighted problem, and it involves increasing the number of vertices in the graph by a factor of $O(\log n)^{\widetilde{O}(t^2)}$. (In contrast, the reduction employed to the vertex-weighted problem increases the number of vertices by a fcator of $O(t^3 \log n)$.)

Given that our reductions increase the number of vertices in the graph, and seeking to maintain that number, we present a reduction of counting t-cliques in $\widetilde{O}(n)$-vertex graphs to counting t-cliques in n-vertex graphs (see Section II-B3). Lastly, given that our worst-case to rare-case reduction reduces to several instance lengths, we also present a reduction from the rare-case problem of counting t-cliques under several distributions to counting them under one distribution (see the full version [18]).

D. Other related work

Several works have constructed interactive (and non-interactive) proof systems for clique-counting, where the interactive proof system of Thaler [28, Apdx] is most related to our work.[9] Specialized to the problem counting t-cliques, this interactive proof system uses $t - 2$ rounds, with $\widetilde{O}(n)$ communication, $O(|E| + n)$ verification time, and $O(|E| \cdot n^{t-2})$ proving time. However, his proof system uses the sum-check protocol as well as the arithmetization approach of [24], which is also followed in [16]. In contrast,

[9] We remark that the protocol of [28] operates in a more challenging streaming setting, which we do not consider or elaborate on in this work.

our proof system (of Theorem 1.3) has the salient feature of deviating from the arithmetization approach of [24], and maintaining the combinatorial flavor of the original problem throughout interaction. Furthermore, the complexity of our prover strategy is directly related to the complexity of counting the number of t-cliques in a graph. In particular, for fixed $t \geq 3$, known algorithms for this problem give a prover runtime of $\widetilde{O}(n^{1+\omega_{\mathtt{mm}} \cdot \lceil (t-1)/3 \rceil}) \ll |E| \cdot n^{t-2}$, where $\omega_{\mathtt{mm}}$ is the matrix multiplication exponent.

While it is unknown whether non-deterministic algorithms (equiv., $\mathcal{NP}$ proof systems) can outperform the best algorithm known for counting t-cliques, Williams [32] showed that such an improvement is possible when allowing randomization; that is, non-interactive and randomized proof systems (as captured by the complexity class $\mathcal{MA}$) can outperform algorithms. Specifically, his (single-message) proof system for counting the number of t-cliques has proof length and verification time $\widetilde{O}(n^{\lfloor t/2 \rfloor + 2}) \ll n^{0.666t}$. Subsequent improvement by Björklund and Kaski [7] yields an $\mathcal{MA}$ proof system with length and verification time $\widetilde{O}(n^{(\omega_{\mathtt{mm}}+\epsilon) \cdot t/6})$, where $\epsilon > 0$ is an arbitrarily small constant. The time to construct proofs in their system is $\widetilde{O}(n^{(\omega_{\mathtt{mm}}+\epsilon)t/3})$, matching the best algorithm known for solving the problem. Recall, however, that we construct *interactive* proof systems: this lets us reduce the verification time to $\widetilde{O}(n^2)$ and the communication to $\widetilde{O}(n)$.

E. Notation and organization

For a natural number n, we let $[n] = \{1, ..., n\}$ and $[[n]] = \{0\} \cup [n]$. For a set U, we let $\binom{U}{t} = \{S \subseteq U : |S| = t\}$. For a prime p, we let $\mathrm{GF}(p)$ denote the finite field of cardinality p. We often conduct our discussion with reference to a seemingly fixed number of vertices (denoted n), clique size (denoted t), and prime p; the reader should think of n as varying (or generic), and of t and p as possibly depending on n.

Organization. The remainder of this extended abstract provides additional details about the worst-case to average-case reductions for counting t-cliques in graphs, using the notion of edge-weighted t-cliques as a methodological vehicle. See the full version [18] for full details, as well as a full exposition about the worst-case to rare-case reductions, direct interactive proofs (in particular, we mention that the interactive proofs use the notion of vertex-weighted t-cliques as a methodological vehicle), and further extensions and related discussions.

II. On counting edge-weighted t-cliques

We consider a generalization of the t-clique counting problem, in which *one is given a graph $G = ([n], E)$ along with edge-weights, and is required to output the sum of the weights of all t-cliques in G*, where the weight of a set $S \subseteq [n]$ is defined as the *product* of the weights of edges between its elements. Given that we use edge weights, there is no need to specify a graph since non-edges can be represented as edges with weight zero. The weight of a t-subset of vertices will be defined as equal the product of the weights of all edges in the induced graph (including the self-loops). Hence, for a symmetric (and possibly reflexive) n-by-n matrix $W = (w_{j,k})_{j,k \in [n]}$, we let $\mathtt{CWC}_t(W)$ denote the sum of the weights of all t-subsets of vertices; that is,

$$\mathtt{CWC}_t(W) \stackrel{\text{def}}{=} \sum_{S \in \binom{[n]}{t}} \prod_{j \leq k : j,k \in S} w_{j,k} \tag{1}$$

Computing this quantity is tightly related to counting the number of t-cliques in simple unweighted graphs. In one direction, for a simple graph $G = ([n], E)$, taking $W_{j,j} = 1$ for every $j \in [n]$, and $W_{j,k} = 1$ if $\{j,k\} \in E$ and $W_{j,k} = 0$ otherwise (i.e., if $\{j,k\} \in \binom{[n]}{2} \setminus E$), the quantity $\mathtt{CWC}_t(W)$ equals the number of t-cliques in G. In the other direction, we shall show how to reduce the computation of $\mathtt{CWC}_t$ to counting t-cliques in graphs, (the reduction, which is presented in Section II-A), incurs a significant overhead in terms of the parameter t).

We also consider the restriction and reduction of $\mathtt{CWC}_t$ to prime fields; that is, abusing notation, for a prime p, we let $\mathtt{CWC}^p_t(W) \stackrel{\text{def}}{=} \mathtt{CWC}_t(W) \bmod p$.

Our motivation for this generalization is that it supports a worst-case to average-case reduction, which will be used to prove Theorem 1.1. But before describing this reduction, we show that the quantity supports a downward self-reduction (in terms of the parameter t), which proves useful for several of our results:

$$t \cdot \mathtt{CWC}_t(W) = \sum_{i \in [n]} w_{i,i} \cdot \mathtt{CWC}_{t-1}(W^{(i)}), \tag{2}$$

where $W^{(i)} = (w^{(i)}_{j,k})$ satisfies:

$$w^{(i)}_{j,k} = \begin{cases} 0 & \text{if } j = k = i \\ w_{i,j} \cdot w_{j,j} & \text{if } j = k \in [n] \setminus \{i\} \\ w_{j,k} & \text{otherwise (i.e., } j \neq k) \end{cases}$$

Note that $W^{(i)}$ and W differ only on the self-loops (i.e., $w^{(i)}_{j,k} = w_{j,k}$ for every $j \neq k$), where $w^{(i)}_{i,i} = 0$ and $w^{(i)}_{j,j} = w_{i,j} \cdot w_{j,j}$ for every $j \in [n] \setminus \{i\}$. To see that Eq. (2) holds, note that each pair (S, i) such that S is a t-subset of $[n]$ and $i \in S$ contributes equally to each side of Eq. (2), where the contribution is $\left(\prod_{j \in S} w_{i,j}\right) \cdot \prod_{j \leq k \in S \setminus \{i\}} w_{j,k}$, which equals $w_{i,i} \cdot \prod_{j \leq k \in S \setminus \{i\}} w^{(i)}_{j,k}$, since both expressions equal $w_{i,i} \cdot \left(\prod_{j \in S \setminus \{i\}} w_{i,j} \cdot w_{j,j}\right) \cdot \prod_{j < k \in S \setminus \{i\}} w_{j,k}$.

A. Reducing counting edge-weighted cliques to the unweighted case

We show that computing $\mathtt{CWC}_t$ can be reduced to counting the number of t-cliques in simple unweighted graphs. We present the reduction in two explicit steps: first reducing the

magnitude of the weights in the matrix, and then reducing to the case of simple graphs (with no weights).

Proposition 2.1: (reducing the weights in the computation of $\mathtt{CWC}_t$): The computation of $\mathtt{CWC}_t$ for matrices with entries in $[[m]]$ can be reduced in $\widetilde{O}(t^2n^2\log m)$-time to computing $\mathtt{CWC}_t^p$ for primes $p \in [t^2\log(nm), 2t^2\log(nm)]$. The reduction makes less than $t^2\log(nm)$ queries, each referring to an n-by-n matrix.

Proof: We use the Chinese Remainder Theorem, while relying on the fact that $\mathtt{CWC}_t(W)$ resides in the interval $[0, n^t \cdot m^{t^2}]$. Specifically, given a matrix $W \in [[m]]^{n\times n}$, for each prime $p \in [t^2\log(nm), 2t^2\log(nm)]$, we query $\mathtt{CWC}_t^p$ on $W^{(p)} = W \bmod p$ (i.e., $W^{(p)} = (w_{j,k} \bmod p)_{j,k\in[n]}$). ■

While the overhead of the foregoing reduction is polynomial in t and $\log m$, the overhead of the following reduction is exponential in these values. Hence, the following reduction is applied only for small values of t (e.g., $t = O(1)$) and small weights (i.e., small p's).

Theorem 2.2: (reducing $\mathtt{CWC}_t^p$ to counting t-cliques in graphs): The computation of $\mathtt{CWC}_t^p : \mathrm{GF}(p)^{n\times n} \to \mathrm{GF}(p)$ can be reduced in $\widetilde{O}(p^{t^2}n^2)$-time to computing the number of t-cliques in an unweighted graph having $n'' \stackrel{\rm def}{=} \widetilde{O}(2^{t^2\lceil\log_2 p\rceil}\cdot n)$ vertices. Furthermore, $\mathtt{CWC}_t(W) = \mathtt{CWC}_t(T(W))/t!$, where $T : \mathrm{GF}(p)^{n\times n} \to \{0,1\}^{n''\times n''}$ is a $\widetilde{O}(p^{t^2}n^2)$-time computable one-to-one mapping and $T(W)$ is an n''-by-n'' symmetric and reflexive Boolean matrix *(which represents an n''-vertex graph).*
(Note that n'' is intentionally defines as a function of $\lceil\log_2 p\rceil$ rather than as a function of p; hence, the problems of computing $\mathtt{CWC}_t^p$, for all $p \in [2^{\ell(n)-1}, 2^{\ell(n)}]$, are reduced to the same clique counting problem.)

Proof: Viewing the (symmetric) matrix $W \in \mathrm{GF}(p)^{n\times n}$ as an n-vertex (complete) graph $G = ([n], \binom{[n]}{2})$ with vertex and edge weights, we first get rid of the vertex weights. This is done by replacing each vertex (of G) by an independent set of size that equals its weight, which is in $[p]$, and placing complete bipartite graphs between these independent sets with edge weights that equal the weight of the corresponding edge. (That is, the vertex v is replaced by an independent set of size $w_{v,v}$, denoted I_v, and the edge between u and v is replaced by a complete bipartite graph between I_u and I_v such that each edge in this bipartite graph has weight $w_{u,v}$.) Hence, we derive a graph with $\widetilde{n} \stackrel{\rm def}{=} \sum_{v\in[n]} w_{v,v} \leq n' \stackrel{\rm def}{=} n\cdot p$ vertices. Augmenting this graph with $n' - \widetilde{n}$ isolated vertices, we obtain an n'-vertex graph G' with edge weights in $[p]$. We denote the weight of edge $e = \{u,v\}$ by w_e (whereas all vertex weights are set to 1). Actually, we also assign weights (of 0) to non-edges; that is, $w_e = 0$ if e is not an edge in G'.

Next, we construct a graph G'' that consists of $p^{\binom{t}{2}}$ isolated copies of graphs of the form $G''_{\bar{i}}$, where $\bar{i} = (i_{j,k})_{j<k\in[t]} \in [p]^{\binom{t}{2}}$, and each graph $G''_{\bar{i}}$ consists of t independent sets such that each pair of sets is connected by a *subgraph* of the double-cover of G'. (Recall that the double-cover of $G' = ([n'], E')$ is a bipartite graph B' with n' vertices on each side such that $\langle 1,u\rangle$ and $\langle 2,v\rangle$ are connected in B' if and only if $\{u,v\} \in E'$.) Specifically, for each $\bar{i} = (i_{j,k})_{j<k\in[t]} \in [p]^{\binom{t}{2}}$, the graph $G''_{\bar{i}}$ consists of the vertex-set $\{\langle \bar{i}, j, v\rangle : j \in [t] \,\&\, v \in [n']\}$ such that vertices $\langle \bar{i}, j, v\rangle$ and $\langle \bar{i}, k, u\rangle$ are connected if and only if ($j \neq k$ and) $w_{\{v,u\}} \geq i_{j,k}$. We stress that the crux of the construction is that, for $j \neq k$, the vertices $\langle \bar{i}, j, v\rangle$ and $\langle \bar{i}, k, u\rangle$ are connected if and only if $i_{j,k} \in [w_{\{v,u\}}]$.

Note that in the case that all weights equal p (i.e., $w_{\{v,u\}} = p$ for every $\{u,v\} \in E'$), each graph $G''_{\bar{i}}$ consists of t independent sets that are connected by double-covers of G'. In this case, each t-clique in G' yields $t!$ cliques of size t in each $G''_{\bar{i}}$, where these $t!$ cliques correspond to all possible permutations over $[t]$; specifically, for each clique $\{v_1, ..., v_t\}$ in G' and each permutation π over $[t]$, the set $\{\langle \bar{i}, \pi(1), v_1\rangle, ..., \langle \bar{i}, \pi(t), v_t\rangle\}$ is a clique in $G''_{\bar{i}}$. On the other hand, the only t-cliques in $G''_{\bar{i}}$ are t-subsets of the form $\{\langle \bar{i}, 1, v_1\rangle, ..., \langle \bar{i}, t, v_t\rangle\}$ such that $\{v_1, ..., v_t\}$ is a t-clique in G'. In the general case (of arbitrary edge weights $(w_e)_{e\in E'} \in [p]^{|E'|}$), for each permutation π over $[t]$, each t-clique $\{v_1, ..., v_t\}$ in G', yields the clique $\{\langle \bar{i}, \pi(1), v_1\rangle, ..., \langle \bar{i}, \pi(t), v_t\rangle\}$ in $G''_{\bar{i}}$ if and only if for all $j < k \in [t]$ it holds that $i_{\pi(j),\pi(k)} \in [w_{\{v_j,v_k\}}]$. Hence, for each permutation π, an "image" of the t-clique $\{v_1, ..., v_t\}$ appears in $\prod_{j<k} w_{\{v_j,v_k\}}$ of the graphs $G''_{\bar{i}}$. On the other hand, the only t-cliques in G'' have the form $\{\langle \bar{i}, 1, v_1\rangle, ..., \langle \bar{i}, t, v_t\rangle\}$ such that $\{v_1, ..., v_t\}$ is a t-clique in G' and $i_{j,k} \in [w_{\{v_j,v_k\}}]$ holds for all $j \neq k \in [t]$.

To summarize, denoting by $W' = (w'_{j,k})$ the matrix that corresponds to the edge weights in G' (i.e., $w'_{j,j} = 1$ and $w'_{j,k} = w_{\{j,k\}}$ if $\{j,k\}$ is an edge of G' (and $w'_{j,k} = 0$ otherwise)[10]), we have $\mathtt{CWC}_t(W) = \mathtt{CWC}_t(W')$ and $t!\cdot\mathtt{CWC}_t(W')$ equals the number of t-cliques in G''. ■

Remark 2.3: (reducing $\mathtt{CWC}_i^p$ to counting t-cliques, where $i \in [t-1]$): Generalizing Theorem 2.2, for every $i \in [t]$, we obtain a mapping $T^{(i)} : \mathrm{GF}(p)^{n\times n} \to \mathrm{GF}(2)^{n^{(i)}\times n^{(i)}}$ such that $\mathtt{CWC}_i(W) = \mathtt{CWC}_i(T^{(i)}(W))/i!$ and $n^{(i)} = \widetilde{O}(2^{i^2\lceil\log_2 p\rceil}\cdot n)$, where indeed $T^{(t)} = T$ and $n^{(t)} = n''$. Wishing to reduce all $\mathtt{CWC}_i$'s to counting t-cliques in unweighted graphs, we *augment* the graph produced by $T^{(i)}$ with $n'' - n^{(i)} - 2(t-i)$ isolated vertices and a clique of size $t-i$ that is connected by a complete biparitite graph to all original vertices (in the graph produced by $T^{(i)}$). Observing that the original graph produced by $T^{(i)}$ has no cliques of size greater than i, it follows that there is a one-to-one correspondance between the i-cliques in the original graph and the t-cliques in the

[10] Recall that G' consists of n independent sets that are connected by complete bipartite graphs (and $n' - \widetilde{n}$ isolated vertices).

augmented graph. We denote the augmented graph derived from W when wishing to compute $\mathtt{CWC}_i(W)$ by $T'(i,W)$, and note that $T'(i,W)$ has $n''-(t-i)$ vertices.

B. The worst-case to average-case reduction

We consider the worst-case and average-case problems of computing $\mathtt{CWC}_t^p : \mathrm{GF}(p)^{n\times n} \to \mathrm{GF}(p)$, while allowing t and p to vary with n. While it is natural to assume that $t = t(n)$ is easily determined given n, we cannot make this assumption regarding the determination of a prime $p = p(n)$ of size $\Theta(n^t)$, since determining such a prime may take time $\Omega(n^t)$, whereas our focus here is on reductions that run much faster (i.e., they must be certainly faster than the complexity of computing $\mathtt{CWC}_t$).[11] Indeed, we can resolve the problem by adopting some form of Cramer's conjecture (which asserts that the interval $[m, m+O(\log^2 m)]$ contains a prime), but prefer not to make such assumptions.[12]

For starters, we shall ignore the foregoing issue, and consider the problem Π_n^p of computing $\mathtt{CWC}_t^p : \mathrm{GF}(p)^{n\times n} \to \mathrm{GF}(p)$, where n, p and t are viewed as generic (and are given to all algorithms as auxiliary inputs). We shall later consider the problem Π_n in which the instances are pairs of the form (p, W) such that p is an $\ell(n)$-bit long prime and $W \in \mathrm{GF}(p)^{n\times n}$, and the problem is to compute $\mathtt{CWC}_{t(n)}^p(W)$.

1) The reduction of $\mathtt{CWC}_t^p$ for fixed t and p: For sake of asymptotic presentation, we let $t = t(n)$ and $p = p(n)$ be functions of n. Recall that we assume that the reductions asserted next are given n, t and p as auxiliary inputs.

Theorem 2.4: (worst-case to average-case reduction for Π_n^p): Fixing functions $t : \mathbb{N} \to \mathbb{N}$ and $p : \mathbb{N} \to \mathbb{N}$ such that $p(n)$ is a prime number in $[\omega(t(n)^2), n^{O(t(n))}]$, we let Π_n^p denote the problem of computing $\mathtt{CWC}_{t(n)}^{p(n)}(W)$ for n-by-n matrices W over $\mathrm{GF}(p(n))$. Then, Π_n^p is randomly self-reducible in time $\widetilde{O}(t(n)^3 \cdot n^2)$ with $t(n)^2$ queries.[13] Furthermore, there is a worst-case to average-case reduction of Π_n^p to itself that makes $O(t(n)^2)$ queries, runs in $\widetilde{O}(t(n)^3 \cdot n^2)$ time, and outputs the correct value with probability $2/3$, provided that the error rate of the average-case solver is a constant smaller than one half.

[11] Indeed, this issue is less acute in the context of interactive proofs, since we may instruct the parties to select such a prime at random (rather than determine it).

[12] Indeed, for our purposes, it suffices to assume that interval $[m, m + \mathrm{poly}(\log m)]$ contains a prime. Actually, we can get meaningful results even when only assuming that interval $[m, m + m^{o(1)}]$ contains a prime.

[13] Recall that a problem is is randomly self-reducible in time T with q queries if there exists an oracle machine of time complexity T and query complexity q that solves the problem (in the worst-case) by making uniformly distributed queries to the problem itself. (We stress that the queries may depend on one another; it is only required that each query is uniformly distributed among the problem's instances.) Indeed, such a reduction yields a worst-case to average-case reduction that supports average-case error rate of $1/3q$.

For every fixed t, the computation of $\mathtt{CWC}_t^p$ can be reduced in $\widetilde{O}(n^2)$-time to computing the number of t-cliques in unweighted $\widetilde{O}(n)$-vertex graphs (see the full version [18] for details). Hence, for $t = O(1)$ and $p = \mathrm{poly}(\log n)$, the worst-case complexity of $\mathtt{CWC}_t^p : \mathrm{GF}(p)^{n\times n}\tau\mathrm{GF}(p)$ is upper bounded by the average-case complexity of counting t-cliques in unweighted $\widetilde{O}(n)$-vertex graphs that are uniformly distributed over the set $\{T(W) : W \in \mathrm{GF}(p)^{n\times n}\}$, where T is the mapping presented in Theorem 2.2.

Proof: Fixing $t = t(n) > 1$ and $p = p(n)$, we let $\mathcal{F} = \mathrm{GF}(p(n))$. For any $W, R \in \mathcal{F}^{n\times n}$, consider the univariate polynomial $P_t(z) = \mathtt{CWC}_t^p(W + zR)$, where the arithmetic is over $\mathcal{F}$. Recalling Eq. (1), observe that P_t has degree $t + \binom{t}{2} < t^2$, and so for every W and R, the value of $\mathtt{CWC}_t^p(W)$ can be obtained by querying P_t at t^2 points. Hence, given W, the random self-reducibility process select $R \in \mathcal{F}^{n\times n}$ uniformly at random, and queries the corresponding polynomial at the points $1, ..., t^2$. Note that these queries correspond to t^2 queries to $\mathtt{CWC}_t^p$ such that each query is uniformly distributed in $\mathcal{F}^{n\times n}$ (by virtue of the random R).

Recovery under error rate of $1/9$ is possible by picking a random R, querying the corresponding polynomial on $3t^2$ (non-zero) points, and employing the Berlekamp–Welch algorithm. In this case, with probability at least $2/3$, at least $2t^2$ queries are answered correctly, and the decoder succeeds since the distance of the code is smaller than t^2. (Note that each of these queries corresponds to a collection of n^2 points on n^2 random lines that pass through W at their origin.)

To support an error rate of $\eta < 1/2$ (equiv., a success rate of $0.5+\epsilon$ for $\epsilon = 0.5-\eta$), we pick a random collection of n^2 curves of degree two, denoted $(C_{i,j} : \mathcal{F} \to \mathcal{F})_{i,j\in[n]}$, that pass (at their origin) through the n^2 corresponding entries of W (i.e., $C_{i,j}(0) = w_{i,j}$), and consider the polynomial (of degree at most $2t^2$) that represents the value of $\mathtt{CWC}_t^p(C(z))$, where $C(z) = (C_{i,j}(z))$. In this case, letting $\epsilon = 0.5 - \eta$, with probability at least $2/3$, at least a $0.5 + 0.5 \cdot \epsilon$ fraction of the $q = O(t^2/\epsilon)$ queries are answered correctly, which suffices for correct decoding (since $\epsilon \cdot q$ is larger than $2t^2$). ■

2) The reduction of Π_n (i.e., $\mathtt{CWC}_t^p$ for varying t and p): Here, for efficiently computable functions $t, \ell : \mathbb{N} \to \mathbb{N}$, we consider the problem Π_n in which the instances are pairs of the form (p, W) such that p is an $\ell(n)$-bit long prime and $W \in \mathrm{GF}(p)^{n\times n}$, and the problem is to compute $\mathtt{CWC}_{t(n)}^p(W)$.

Showing a worst-case to average-case reduction for Π_n is more complex than doing so for Π_n^p, because when given the input (p, W) it may be the case that the average-case solver just fails on all inputs of the form $(p, \cdot)$. Hence, we shall use Chinese Remaindering (with errors [15]) in order to obtain the value of $\mathtt{CWC}_{t(n)}^p(W)$ (or rather $\mathtt{CWC}_{t(n)}(W)$) from the values of $\mathtt{CWC}_{t(n)}^{p'}(W)$ for other primes $p' \in [2^{\ell(n)-1}, 2^{\ell(n)}]$.

Theorem 2.5: (worst-case to average-case reduction for Π_n):[14] Let $\ell : \mathbb{N} \to \mathbb{N}$ be such that $\ell(n) \in [3\log t(n) + \log\log n + \omega(1), O(t(n)\log n)]$. Then, there exists a worst-case to average-case reduction of Π_n to itself that makes $\widetilde{O}(t(n)^5 \log n)$ queries, runs in $\widetilde{O}(t(n)^6 \cdot n^2)$ time, and outputs the correct value with probability $2/3$, provided that the error rate of the average-case solver is a constant smaller than one fourth.

Proof: Denoting the error rate of the average-case solver by $\eta < 1/4$, and letting $\epsilon = 1/4 - \eta$, we first observe that for at least a $0.5 + \epsilon$ fraction the primes in $I_n \stackrel{\text{def}}{=} [2^{\ell(n)-1}, 2^{\ell(n)}]$, the solver has an error rate of at most $0.5 - \epsilon$; that is, for each such prime p, hereafter called good, the solver solves Π_n^p correctly on at least a $0.5 + \epsilon$ fraction of the instances. Hence, for each good prime, we can apply the reduction presented in the proof of Theorem 2.4, but the problem is that the prime that is part of the worst-case instance may not be good.

Hence, on input (p, W), where $p \in I_n$ and $W \in \mathrm{GF}(p)^{n\times n}$, we first try to obtain the (integer) value of $\mathtt{CWC}_{t(n)}(W)$, and then reduce the result modulo p. Basically, we shall obtain the value of $\mathtt{CWC}_{t(n)}(W)$ by selecting $m = O(\epsilon^{-1} t(n)^3 \log n)/\ell(n)$ random primes $p_1, ..., p_m$ in I_n, hoping that at least $(0.5 + 0.5\epsilon) \cdot m$ of them are good, and combining the values $(\mathtt{CWC}^{p_i}_{t(n)}(W))_{i\in[m]}$ using Chinese Remaindering with errors [15, Sec. 3]. The analysis of the latter decoding relies on the fact that the (non-negative) value of $\mathtt{CWC}_{t(n)}(W)$ is bounded above by $\binom{n}{t(n)} \cdot (2^{\ell(n)})^{t(n)^2} < n^{O(t(n)^3)}$, whereas the product of the smallest ϵm primes in I_n exceeds $(2^{\ell(n)-1})^{\epsilon m} = \exp(O(t(n)^3 \log n))$.[15] Specifically, on input (p, W), we proceed as follows.

1) Select at random $m = \frac{O(\epsilon^{-1} \cdot t(n)^3 \log n)}{\ell(n)}$ primes in I_n.
2) For each selected prime, denoted p_i, invoke the worst-case to average-case procedure for $\Pi_n^{p_i}$ on the instance (p_i, W), and denote the result by v_i. (The aforementioned reduction is the one presented in the proof of Theorem 2.4, except that the error probability should be reduced to $1/3m$.)[16]
3) Apply Chinese Remaindering with errors (for error rate $0.5 - 0.5\epsilon$) on $v_1, ..., v_m$ (and the priomes $p_1, ..., p_m$), and output the result reduced modulo p.

The theorem follows by using the fact that, with high probability, at least a $0.5 + 0.5\epsilon$ fraction of the primes selected in Step 1 are good. ■

Corollary 2.6: (worst-case to average-case reduction for counting cliques): Let t be a constant and $b(n) = \Theta(t^3 \log n)$. Let $\mathcal{G}_n$ be a distribution on $\widetilde{O}(n)$-vertex graphs obtained by selecting uniformly at random a prime $p \in [b(n), 2b(n)]$ and $W \in \mathrm{GF}(p)^{n\times n}$, and outputting $T(W)$ where T is the mapping presented in Theorem 2.2. Then, there exists a worst-case to average-case reduction of counting t-cliques in n-vertex graphs to counting t-cliques in graphs generated according to $\mathcal{G}_n$ such that the reduction runs in $\widetilde{O}(n^2)$ time, makes $\widetilde{O}(\log n)^2$ queries, and outputs the correct value with probability $2/3$, provided that the error rate of the average-case solver is a constant smaller than one fourth.

Proof: Given an n-vertex graph G, using Proposition 2.1, we reduce counting the number of t-cliques in G to making $O(\log n)$ queries to oracles of the form $\mathtt{CWC}^p_t$ such that p is a prime in $[b(n), 2b(n)]$. Next, setting $\ell(n) = \lceil \log b(n) \rceil$ (and using Theorem 2.5), we reduce answering each of these queries to solving the problem Π_n on at least $0.75+\epsilon$ of the instances, where $\epsilon > 0$ is an arbitrary constant. (We do so after reducing the error probility of the reduction to $o(1/\log n)$.) Lastly, using the mapping T of Theorem 2.2, we map the $\widetilde{O}(t^5 \log n)$ random queries made by the worst-case to average-case reduction to $\widetilde{O}(n)$-vertex graphs. We stress that a procedure that counts t-cliques in $\mathcal{G}_n$ correctly with probability $0.75 + \epsilon$, yields a procedure that answers Π_n correctly with probability $0.75 + \epsilon$. ■

3) Length reduction: Corollary 2.6 falls short from establishing Theorem 1.1 only in one aspect: It reduces worst-case n-vertex graph instances to average-case instances that are n''-vertex graphs, for $n'' = \widetilde{O}(n)$. Wishing to have a worst-case to average-case reduction that preserves the number of vertices (in the corresponding instances), we seek a reduction that reduces the number of vertices in the graph. It seems easiest to present such a reduction in the worst-case setting. Indeed, we show a reduction of counting t-cliques in n-vertex graphs to counting t-cliques in n'-vertex graphs such that $n' = n/\mathrm{poly}(\log n)$. Applying Corollary 2.6 to the resulting n'-vertex graphs, we reduce to n''-vertex graphs such that $n'' = \widetilde{O}(n') = n$.

Proposition 2.7: (reducing the number of vertices in the t-clique counting problem): Let t be a constant and $k : \mathbb{N} \to \mathbb{N}$ such that $k(n) \in [t + 1, n - 1]$. Then, there exists an $O((n/k(n))^t \cdot k(n)^2)$-time reduction of counting t-cliques in n-vertex graphs to counting t-cliques in $k(n)$-vertex graphs. Furthermore, the reduction performs $O(n/k(n))^t$ queries.

Proof: Consider an arbitrary partition of $[n]$ into $m = \lceil t \cdot n/(k(n) - t) \rceil$ sets $V_1, ..., V_m$ such that $|V_i| \leq k(n)/t$ (for every $i \in [m]$). For every $I \subset [m]$ of size at most t, let

[14] Recall that Π_n is defined in terms of of the functions $t, \ell : \mathbb{N} \to \mathbb{N}$. It refers to instances of the form (p, W) such that p is an $\ell(n)$-bit long prime and $W \in \mathrm{GF}(p)^{n\times n}$, and calls for computing $\mathtt{CWC}^p_{t(n)}(W)$.

[15] Specifically, if $\ell(n) \leq c \cdot t(n)\log n$, then letting $m = 2c \cdot (\epsilon^{-1} t(n)^3 \log n)/\ell(n)$ will do, since $\binom{n}{t(n)} \cdot (2^{\ell(n)})^{t(n)^2} < n^{(c+o(1))\cdot t(n)^3}$, whereas $(2^{\ell(n)-1})^{\epsilon m} = 2^{(1-o(1))\cdot 2c\cdot t(n)^3 \log n}$. The unique decoding condition in [15] essentially requires an error rate of $0.5 - \frac{0.5k}{m}$, assuming that the product of the smallest k (out of m) primes exceeds the encoded (non-zero) integer.

[16] Hence, each invokation generates $O(t(n)^2 \log m)$ queries, totaling in $\widetilde{O}(m) \cdot t(n)^2$ queries.

$V_I = \cup_{i \in I} V_i$ and G_I be the subgraph of G induced by V_I. Next, *augment* each G_I to a $k(n)$-vertex graph, denoted G'_I, by possiblly adding $k(n) - |V_I|$ isolated vertices (and note that the t-cliques in G'_I are exactly those in G_I). Observe that for each t-clique of G there exists a unique I (of size at most t) such that this t-clique appears in G_I but does not appear in any $G_{I'}$ such that $I' \subset I$. Letting N_I denote the number of t-cliques that appear in G_I but do not appear in any $G_{I'}$ such that $I' \subset I$, it follows that the number of t-cliques in G equals $\sum_{i \in [t]} \sum_{I \in \binom{[m]}{i}} N_I$. Hence, on input $G = ([n], E)$, the reduction proceeds as follows:

1) For each $I \in \bigcup_{i \in [t]} \binom{[m]}{i}$, it queries for the number of t-cliques in G'_I, denoting the result by r_I.
2) It computes all N_I based on the values obtained in Step 1. Specifically, for $i = 1, ..., t$, and for every $I \in \binom{[m]}{i}$, it sets $N_I \leftarrow r_I - \sum_{I' \subset I} N_{I'}$, where $N_\emptyset = 0$.

The reduction outputs the sum of all the N_I's. Observing that the reduction makes $\sum_{i \in [t]} \binom{m}{i} < m^t$ queries, the claim follows. ∎

Proof of Theorem 1.1: Combining Corollary 2.6 with Proposition 2.7 yields Theorem 1.1. Specifically, for a suitable polynomial p, we set $k(n) = n/p(\log n)$, and reduce counting t-cliques in n-vertex graphs to counting them in $k(n)$-vertex graphs (using Proposition 2.7). Then, we apply Corollary 2.6 with n replaced by $k(n)$, reducing the worst-case problem for $k(n)$-vertex graphs to an average-case problem regarding the distribution $\mathcal{G}_{k(n)}$, which is supported by $\widetilde{O}(k(n))$-vertex graphs. Indeed, a suitable choice of the polynomial p implies that $\widetilde{O}(k(n)) = \widetilde{O}(n)/p(\log n) \leq n$, and Theorem 1.1 follows (possibly by augmenting the graph with isolated vertices). ∎

REFERENCES

[1] Amir Abboud, Kevin Lewi, and Ryan Williams. Losing Weight by Gaining Edges. In *22nd ESA*, pages 1–12, 2014

[2] Miklos Ajtai. Σ^1_1-formulae on finite structures. *Ann. Pure Appl. Logic*, Vol. 24 (1), pages 1–48, 1983.

[3] Laszlo Babai. Random oracles separate PSPACE from the Polynomial-Time Hierarchy. *IPL*, Vol. 26, pages 51–53, 1987.

[4] Laszlo Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP has Subexponential Time Simulations unless EXPTIME has Publishable Proofs. *Complexity Theory*, Vol. 3, pages 307–318, 1993.

[5] Marshall Ball, Alon Rosen, Manuel Sabin and Prashant Nalini Vasudevan. Average-case fine-grained hardness. In *49th ACM Symposium on the Theory of Computing*, pages 483–496, 2017.

[6] Marshall Ball, Alon Rosen, Manuel Sabin and Prashant Nalini Vasudevan. Proofs of Useful Work. IACR Cryptology ePrint Archive, Report 2017/203, 2017.

[7] Andreas Björklund and Petteri Kaski. How Proofs are Prepared at Camelot. In *35th ACM Symposium on Principles of Distributed Computing*, pages 391–400, 2016.

[8] Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-Testing/Correcting with Applications to Numerical Problems. *Journal of Computer and System Science*, Vol. 47 (3), pages 549–595, 1993. Preliminary version in *22nd STOC*, 1990.

[9] Andrej Bogdanov and Luca Trevisan. On Worst-Case to Average-Case Reductions for NP Problems. *SIAM Journal on Computing*, Vol. 36 (4), pages 1119–1159, 2006.

[10] Jin-yi Cai, Aduri Pavan, and D. Sivakumar. On the Hardness of Permanent. In *16th STACS*, pages 90–99, 1999.

[11] Cynthia Dwork and Moni Naor. Pricing via Processing or Combatting Junk Mail. In the proceedings of *CRYPTO*, pages 139–147, 1992.

[12] Joan Feigenbaum and Lance Fortnow. Random-Self-Reducibility of Complete Sets. *SIAM Journal on Computing*, Vol. 22 (5), pages 994–1005, 1993.

[13] Oded Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008.

[14] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that Yield Nothing but their Validity or All Languages in NP Have Zero-Knowledge Proof Systems. *Journal of the ACM*, Vol. 38, No. 3, pages 691–729, 1991. Preliminary version in *27th FOCS*, 1986.

[15] Oded Goldreich, Dana Ron, and Madhu Sudan. Chinese Remaindering with Errors. *IEEE Trans. Information Theory*, Vol. 46 (4), pages 1330–1338, 2000. Preliminary version in *31st STOC*, 1999.

[16] Oded Goldreich and Guy N. Rothblum. Simple Doubly-Efficient Interactive Proof Systems for Locally-Characterizable Sets. *ECCC*, TR17-018, February 2017.

[17] Oded Goldreich and Guy N. Rothblum. Worst-case to Average-case reductions for subclasses of P. *ECCC* TR17-130, 2017.

[18] Oded Goldreich and Guy N. Rothblum. Counting t-cliques: Worst-case to average-case reductions and Direct interactive proof systems. *ECCC*, TR18-046, 2018.

[19] Oded Goldreich and Avi Wigderson. Derandomization that is rarely wrong from short advice that is typically good. *ECCC*, TR02-039, 2002.

[20] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating Computation: Interactive Proofs for Muggles. *Journal of the ACM*, Vol. 62(4), Art. 27:1-27:64, 2015. Extended abstract in *40th STOC*, pages 113–122, 2008.

[21] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM Journal on Computing*, Vol. 18, pages 186–208, 1989. Preliminary version in *17th STOC*, 1985. Earlier versions date to 1982.

[22] Russell Impagliazzo and Avi Wigderson. Randomness vs Time: Derandomization under a Uniform Assumption. *Journal of Computer and System Science*, Vol. 63 (4), pages 672–688, 2001.

[23] Richard J. Lipton. New directions in testing. *Distributed Computing and Cryptography*, J. Feigenbaum and M. Merritt (ed.), DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematics Society, Vol. 2, pages 191–202, 1991.

[24] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM*, Vol. 39, No. 4, pages 859–868, 1992. Extended abstract in *31st FOCS*, 1990.

[25] Or Meir. IP = PSPACE Using Error-Correcting Codes. *SIAM Journal on Computing*, Vol. 42 (1), pages 380–403, 2013.

[26] Omer Reingold, Guy N. Rothblum, Ron D. Rothblum. Constant-round interactive proofs for delegating computation. In *48th ACM Symposium on the Theory of Computing*, pages 49–62, 2016.

[27] Adi Shamir. IP = PSPACE. *Journal of the ACM*, Vol. 39, No. 4, pages 869–877, 1992. Preliminary version in *31st FOCS*, 1990.

[28] Justin Thaler. Semi-Streaming Algorithms for Annotated Graph Streams. In *43rd International Colloquium on Automata, Languages, and Programming*, pages 59:1–59:14, 2016.

[29] Madhu Sudan, Luca Trevisan, and Salil P. Vadhan. Pseudorandom Generators without the XOR Lemma. *Journal of Computer and System Science*, Vol. 62 (2), pages 236–266, 2001.

[30] Leslie G. Valiant. The Complexity of Computing the Permanent. Theoretical Computer Science, Vol. 8, pages 189–201, 1979.

[31] Virginia Vassilevska Williams. Hardness of Easy Problems: Basing Hardness on Popular Conjectures such as the Strong Exponential Time Hypothesis. In *10th International Symposium on Parameterized and Exact Computation*, pages 17–29, 2015.

[32] Ryan Williams. Strong ETH Breaks With Merlin and Arthur: Short Non-Interactive Proofs of Batch Evaluation. In *31st Conference on Computational Complexity*, pages 2:1–2:17, 2016.