Improved Merlin-Arthur Protocols for Central Problems in Fine-Grained Complexity[*]

Shyan Akmal
MIT
naysh@mit.edu

Lijie Chen
MIT
lijieche@mit.edu

Ce Jin
MIT
cejin@mit.edu

Malvika Raj
UC Berkeley
malvika@berkeley.edu

Ryan Williams
MIT
rrw@mit.edu

Abstract

In a Merlin-Arthur proof system, the proof verifier (Arthur) accepts valid proofs (from Merlin) with probability 1, and rejects invalid proofs with probability arbitrarily close to 1. The running time of such a system is defined to be the length of Merlin's proof plus the running time of Arthur. We provide new Merlin-Arthur proof systems for some key problems in fine-grained complexity. In several cases our proof systems have optimal running time. Our main results include:

- Certifying that a list of n integers has no 3-SUM solution can be done in Merlin-Arthur time $\tilde{O}(n)$. Previously, Carmosino et al. [ITCS 2016] showed that the problem has a nondeterministic algorithm running in $\tilde{O}(n^{1.5})$ time (that is, there is a proof system with proofs of length $\tilde{O}(n^{1.5})$ and a deterministic verifier running in $\tilde{O}(n^{1.5})$ time).
- Counting the number of k-cliques with total edge weight equal to zero in an n-node graph can be done in Merlin-Arthur time $\tilde{O}(n^{\lceil k/2 \rceil})$ (where $k \geq 3$). For odd k, this bound can be further improved for sparse graphs: for example, counting the number of zero-weight triangles in an m-edge graph can be done in Merlin-Arthur time $\tilde{O}(m)$. Previous Merlin-Arthur protocols by Williams [CCC'16] and Björklund and Kaski [PODC'16] could only count k-cliques in unweighted graphs, and had worse running times for small k.
- Computing the All-Pairs Shortest Distances matrix for an n-node graph can be done in Merlin-Arthur time $\tilde{O}(n^2)$. Note this is optimal, as the matrix can have $\Omega(n^2)$ nonzero entries in general. Previously, Carmosino et al. [ITCS 2016] showed that this problem has an $\tilde{O}(n^{2.94})$ nondeterministic time algorithm.
- Certifying that an n-variable k-CNF is unsatisfiable can be done in Merlin-Arthur time $2^{n/2-n/O(k)}$. We also observe an algebrization barrier for the previous $2^{n/2} \cdot \mathrm{poly}(n)$-time Merlin-Arthur protocol of R. Williams [CCC'16] for #SAT: in particular, his protocol algebrizes, and we observe there is no algebrizing protocol for k-UNSAT running in $2^{n/2}/n^{\omega(1)}$ time. Therefore we have to exploit non-algebrizing properties to obtain our new protocol.
- Certifying a Quantified Boolean Formula is true can be done in Merlin-Arthur time $2^{4n/5} \cdot \mathrm{poly}(n)$. Previously, the only nontrivial result known along these lines was an Arthur-Merlin-Arthur protocol (where Merlin's proof depends on some of Arthur's coins) running in $2^{2n/3} \cdot \mathrm{poly}(n)$ time.

Due to the centrality of these problems in fine-grained complexity, our results have consequences for many other problems of interest. For example, our work implies that certifying there is no Subset Sum solution to n integers can be done in Merlin-Arthur time $2^{n/3} \cdot \mathrm{poly}(n)$, improving on the previous best protocol by Nederlof [IPL 2017] which took $2^{0.49991n} \cdot \mathrm{poly}(n)$ time.

[*] Supported by NSF CCF-1909429, NSF CCF-2127597, and NSF CCF-2129139. Shyan Akmal was partially supported by a Siebel Scholarship. Lijie Chen was also partially supported by an IBM Fellowship. Malvika Raj was supported by a Fano Undergraduate Research and Innovation Scholarship at MIT.

ISSN 1433-8092

1 Introduction

Fine-grained complexity has identified core problems that act as bottlenecks to obtaining faster algorithms for various tasks in computer science. Perhaps the most prominent problems among these are Satisfiability (SAT), Orthogonal Vectors (OV), 3-SUM, and All-Pairs Shortest Paths (APSP). The hypotheses that the known algorithms for these problems are essentially optimal have led to far-reaching consequences for the exact complexity of many problems of interest (see for example the survey by Vassilevska Williams [Vas18]). There is now a vast web of "fine-grained" reductions among computational tasks, which has led to large equivalence classes of problems [Păt10, VW18, LVW18, AGV15, BDT16, GIKW19, CW19], many of which *a priori* look unrelated. For each of these equivalence classes, solving one problem in the class faster means that all problems in that class have more efficient algorithms.

Recently there has been growing interest in obtaining efficient Merlin-Arthur (MA) proof systems for problems studied in fine-grained complexity. Recall that in a Merlin-Arthur proof system, the probabilistic verifier (Arthur) always accepts valid proofs (from the prover Merlin), and rejects invalid proofs with probability arbitrarily close to 1. Williams [Wil16] shows (among other results) that the OV problem[1] for sets of n vectors with dimension d can be solved by a Merlin-Arthur protocol in near-optimal $\tilde{O}(nd)$ time,[2] achieving a nearly quadratic speedup compared to the fastest known algorithms for OV [AWY15, CW21]. One consequence of this result is the refutation of the Merlin-Arthur Strong Exponential-Time Hypothesis, which could be viewed as evidence against the Nondeterministic Strong Exponential-Time Hypothesis (NSETH) proposed by Carmosino et al. [CGI$^+$16].[3]

The main technical component in Williams' work is a "batch evaluation" protocol for low-degree arithmetic circuits, with which Merlin can quickly convince Arthur of the outputs of a circuit C on a set of points $a_1, \ldots, a_K$, faster than evaluating C independently on each point a_i. This protocol can be used to obtain efficient Merlin-Arthur protocols for various other problems, such as #SAT, counting dominating pairs of vectors, and counting Hamiltonian cycles [Wil16]. Building on Williams' batch-evaluation protocol and employing additional algebraic techniques, Björklund and Kaski [BK16] obtained improved Merlin-Arthur protocols[4] for more problems, such as #k-Clique, Graph Coloring, and Set Cover. For many of these problems, the obtained Merlin-Arthur protocols achieve quadratic speedup compared to the fastest known algorithms. Variants of these protocols have been used as Proofs of Work based on fine-grained hardness assumptions [BRSV18], which have led to further work in fine-grained cryptography and average-case fine-grained complexity [BRSV17, GR18a, BBB19, LLV19, GR20b, DLV20, HS21]. In [GR18b, GR20a], doubly efficient proof systems for #k-SUM, #k-Clique, and APSP were constructed, in which the prover runs in polynomial time (for constant k) and the verifier runs in "almost linear" time (i.e., $N^{1+o(1)}$ time, where N is the input length).[5] Efficient batch verification using interactive protocols with a constant (but greater than two) number of rounds has also been developed for problems with polynomial-time verifiable, unique witnesses [RRR18] and problems which can be solved by algorithms with small space complexity [RRR21]. A different line of work in the stream verification setting has developed sublinear space protocols for various graph problems [CG19, CGT20].

Given the interest in fine-grained complexity and proof systems, a natural question is to understand the Merlin-Arthur time complexity of core problems in fine-grained complexity. How efficiently can solutions to these problems be *verified*, with a randomized verifier? As seen above, such questions may have cryptographic applications, and in general they may give insight into the structure of these problems. Williams [Wil16] already showed that OV admits near-optimal Merlin-Arthur protocols. In this work, we present improved Merlin-Arthur protocols for 3-SUM, APSP, and many more core computational tasks. For several of these problems our protocols yield optimal running times (up to polylogarithmic factors) for the verifier.

[1] The OV problem is the following: Let $d = \omega(\log n)$; given two sets $A, B \subseteq \{0,1\}^d$ with $|A| = |B| = n$, determine whether there exist $a \in A$ and $b \in B$ so that $\sum_{i=1}^{d} a_i \cdot b_i = 0$.

[2] We use $\tilde{O}(f(n))$ to hide polylog$(f(n))$ factors.

[3] Informally, NSETH says that unsatisfiable CNFs on n variables require proofs of length at least $2^{n-\varepsilon n}$ for all $\varepsilon > 0$. More formally, NSETH states that for every $\varepsilon > 0$ there exists k such that unsatisfiability of k-CNF formulas cannot be decided in nondeterministic $2^{(1-\varepsilon)n}$ time.

[4] Björklund and Kaski [BK16] actually work in a more restricted setting they call "Camelot algorithms," where Merlin's proof can be prepared efficiently by a parallel algorithm.

[5] The protocols for #k-SUM and #k-Clique have $O(k \log n)$ rounds (or $O(k/\varepsilon)$ rounds at the cost of making verifier running time $n^{1+\varepsilon}$) and the protocol for APSP has constant rounds. We also remark that in our Merlin-Arthur protocols from Theorem 1.1 and Theorem 1.5, the prover runs in polynomial time for constant k as well.

1.1 Our Results

In this section, we describe our new results and compare them with previous work.

Faster Protocols for k-SUM and Related Problems. In the k-SUM problem, we are given n integers from $[-n^c, n^c]$ for some constant c (only depending on k), and wish to determine if some k of them sum to zero. The unparameterized version of this problem is the Subset Sum problem, in which we are given n positive integers less than or equal to 2^{cn}, for some constant c, and a target integer t, and must decide if some subset of the input integers sums to t.

Our first result is a Merlin-Arthur protocol for certifying there is no k-SUM solution, which is significantly more efficient than the best known nondeterministic algorithm [CGI+16] for the same task, which runs in $\tilde{O}(n^{k/2})$ time.[6]

Theorem 1.1. *For any fixed integer $k \geq 3$, certifying that a list of n integers has no k-SUM solution can be done in Merlin-Arthur time $\tilde{O}(n^{k/3})$.*

In particular, our protocol for 3-SUM runs in near-optimal $\tilde{O}(n)$ time. As an immediate corollary, we obtain a faster Merlin-Arthur protocol for certifying a Subset Sum instance has no solution.

Corollary 1.2. *Certifying that a list of n integers has no Subset Sum solution can be done in $2^{n/3} \cdot \mathrm{poly}(n)$ Merlin-Arthur time.*

The previous best Merlin-Arthur protocol for Subset Sum was presented by Nederlof in [Ned17] and takes $2^{0.49991n} \cdot \mathrm{poly}(n)$ time.[7] Note that Subset Sum can be solved deterministically in $O(2^{n/2})$ time [HS74].

In the MinPlus Convolution problem, we are given two integer arrays

$$a = (a_0, a_1, \ldots, a_{n-1}) \quad \text{and} \quad b = (b_0, b_1, \ldots, b_{n-1}),$$

and want to compute the array c whose entries are defined by taking

$$c_k = \min_{i+j=k} \{a_i + b_j\} \qquad (\text{for } 0 \leq k \leq 2n-2).$$

The best known algorithm for MinPlus Convolution takes $n^2/2^{\Omega(\sqrt{\log n})}$ time [Wil18, BCD+14, CW21]. It is known that MinPlus Convolution has a fine-grained reduction to 3-SUM [VW13, Păt10, CMWW19]. We observe that these reductions combined with Theorem 1.1 imply a near-linear time Merlin-Arthur protocol for MinPlus Convolution.

Corollary 1.3. *MinPlus Convolution can be solved in Merlin-Arthur time $\tilde{O}(n)$.*

The All-Numbers k-SUM problem is a seemingly harder version of k-SUM, where we want to decide for every input integer whether or not it belongs to a set of k inputs that sum to zero. It is known that All-Numbers k-SUM and k-SUM are fine-grained equivalent [VW18, Theorem 8.1] in the sense that one of the problems can be solved in $O(n^{\lceil k/2 \rceil - \varepsilon})$ time for some $\varepsilon > 0$ if and only if the other problem can also be solved in that running time, but with possibly a *different* ε. We observe that our k-SUM protocol can be extended to solve All-Numbers k-SUM in the same running time.

Corollary 1.4. *For any fixed integer $k \geq 3$, the All-Numbers k-SUM problem can be solved in Merlin-Arthur time $\tilde{O}(n^{k/3})$.*

[6] Carmosino et al. [CGI+16] gave a co-nondeterministic algorithm for 3-SUM running in $\tilde{O}(n^{3/2})$ time. It is straightforward to extend their algorithm to nondeterministically certify an instance of k-SUM has no solution in $\tilde{O}(n^{k/2})$ time for all $k \geq 3$.

[7] In the arXiv version of [Ned17], it was claimed that combining [Wil16] and [AKKN16] would yield a Merlin-Arthur protocol in $2^{0.3113n} \cdot \mathrm{poly}(n)$ time. The author later confirmed that this claim had a bug [Ned21].

Counting zero-weight cliques. In the #Zero-Weight k-Clique problem, we are given a simple undirected graph G on n vertices and m edges with integer edge weights from $[-n^c, n^c]$ for a positive constant c, and are tasked with counting the number of k-cliques in G whose edge weights sum to zero. The easier #k-Clique problem is equivalent to the special case of #Zero-Weight k-Clique where all edge weights in the input graph are zero.

These two problems have been extensively studied in fine-grained complexity. The trivial brute-force algorithm for both problems runs in $O(n^k)$ time. The #k-Clique problem can be solved faster using fast matrix multiplication [IR78, NP85]. For the detection versions of these problems, it has been conjectured that Zero-Weight k-Clique cannot be solved faster than $n^{k-\varepsilon}$, and that k-Clique cannot be solved faster than $n^{k\omega/3-\varepsilon}$ (where $2 \leq \omega < 2.373$ denotes the matrix multiplication exponent [Vas12, LG14, AV21]), for any constant $\varepsilon > 0$ [ABV18, LVW18]. Some recent works employ even stronger conjectures about the hardness of k-Clique for integers k not divisible by 3 [AGI$^+$19]. For $k = 3$, the Zero-Weight k-Clique problem is simply the Zero-Weight Triangle problem, and it is known that any truly subcubic algorithm for this problem would refute both the APSP conjecture and the 3-SUM conjecture [VW13, VW18].

It is known that #k-Clique can be solved faster using Merlin-Arthur protocols. Williams [Wil16] showed that #k-Clique can be solved in Merlin-Arthur time $\tilde{O}(n^{\lfloor k/2 \rfloor + 2})$. This was later improved by Björklund and Kaski's [BK16] protocol which solves #k-Clique (for integers k divisible by 6) in Merlin-Arthur time $\tilde{O}(n^{k\omega/6})$, achieving a quadratic speedup compared to the best known algorithm in $\tilde{O}(n^{k\omega/3})$ time [IR78, NP85].

We present an improved Merlin-Arthur protocol for the harder #Zero-Weight k-Clique problem.

Theorem 1.5. *For any fixed integer $k \geq 3$, #Zero-Weight k-Clique on a graph with n nodes and m edges can be solved by a Merlin-Arthur protocol with proof length $\tilde{O}(n^{\lfloor k/2 \rfloor})$ and verification time $\tilde{O}\big(n^{\lceil k/2 \rceil} \cdot \big(m/n^2\big)^{\lfloor (k+1)/4 \rfloor} + n^{\lfloor k/2 \rfloor}\big)$.*

For even integers $k \geq 4$ the protocol has proof length and verification time $\tilde{O}(n^{k/2})$, and for odd $k \geq 3$ the protocol has proof length $\tilde{O}(n^{(k-1)/2})$ and verification time

$$\tilde{O}(n^{(k+1)/2} \cdot \big(m/n^2\big)^{\lfloor (k+1)/4 \rfloor} + n^{(k-1)/2}\big) \leq \tilde{O}(m^{\lceil k/4 \rceil} + n^{(k-1)/2}).$$

Two notable cases are $k = 3$ and $k = 4$, for which Theorem 1.5 shows that #Zero-Weight 4-Clique can be solved in Merlin-Arthur time $\tilde{O}(n^2)$ which is near-optimal for dense graphs, and #Zero-Weight Triangle can be solved with proof length $\tilde{O}(n)$ and verification time $\tilde{O}(m)$, which is near-optimal for graphs of any sparsity. Applying known reductions [VW18] immediately implies quadratic time Merlin-Arthur protocols for the following problems, which we define later in Section 4.

Corollary 1.6. *MinPlus Product, APSP, and #Negative Triangle can be solved in Merlin-Arthur time $\tilde{O}(n^2)$.*

Unsatisfiability of k-CNFs. We give a Merlin-Arthur protocol for certifying the unsatisfiability of a k-CNF (*i.e.*, solving the k-UNSAT problem), which runs faster than the previously known $2^{n/2} \cdot \text{poly}(n)$-time protocol.

Theorem 1.7. *There is a universal constant $\delta > 0$ such that for all sufficiently large integers $k > 0$, we can verify any unsatisfiable n-variable m-clause k-CNF with a Merlin-Arthur protocol running in $2^{n(1/2-\delta/k)} \cdot \text{poly}(n, m)$ time.*

Previously, Williams [Wil16] had shown it is possible to count the number of satisfying assignments to CNFs on n variables and m clauses with a Merlin-Arthur protocol in $2^{n/2} \cdot \text{poly}(n, m)$ time. We find Theorem 1.7 intriguing, not just because it runs more efficiently, but also because the result provably must not algebrize (in the sense of [AW09, IKK09]). In particular, we observe that

- Williams' Merlin-Arthur protocol for #SAT algebrizes, and
- there is no algebrizing protocol for k-UNSAT running in $2^{n/2}/n^{\omega(1)}$ time.

More formally, we have the following two theorems:

Proposition 1.8 (Williams' Protocol Algebrizes)**.** *For every oracle A, $\#\mathsf{CNF\text{-}SAT}^A$ on formulas with n variables and size* $\mathrm{poly}(n)$ *can be computed in Merlin-Arthur time* $2^{n/2} \cdot \mathrm{poly}(n)$ *with oracle access to the multilinear extension of A over any field of characteristic greater than 2^n (and order at most $2^{\mathrm{poly}(n)}$).*

Proposition 1.9 (Follows from [AW09])**.** *There is an oracle A such that there is* **no** *Merlin-Arthur protocol running in $2^{n/2}/n^{\omega(1)}$ time for $\mathsf{1\text{-}UNSAT}^A$, even for protocols with oracle access to the multilinear extension of A (over any field of order $2^{\mathrm{poly}(n)}$).*

Therefore the properties exploited in the protocol of Theorem 1.7 (which applies an earlier reduction of [IP01] from fine-grained complexity) are provably non-algebrizing. We are hopeful that further study of such results may lead to new progress in lower bounds via algorithms.

Quantified Boolean Formulas. A Quantified Boolean Formula in prenex normal form (QBF), is a formula

$$(Q_1 x_1) \cdots (Q_n x_n)\ F(x_1, \ldots, x_n),$$

consisting of a propositional formula F of size m over n variables, preceded by quantifiers $Q_i \in \{\exists, \forall\}$. Deciding whether a given QBF is true is a canonical PSPACE-complete problem.

Williams [Wil16] gave a 3-round interactive protocol (*i.e.*, an AMA protocol) for true QBFs that runs in $2^{2n/3} \cdot \mathrm{poly}(n, m)$ time. The same work [Wil16] raised the question of whether there is a Merlin-Arthur (two-round) protocol for deciding true QBFs which runs in $2^{(1-\varepsilon)n} \cdot \mathrm{poly}(n, m)$ time for some constant $\varepsilon > 0$. We resolve this open problem in the affirmative:

Theorem 1.10. *True Quantified Boolean Formulas (TQBF) with n variables and size $m \leq 2^n$ can be certified by a Merlin-Arthur protocol running in* $2^{4n/5} \cdot \mathrm{poly}(n, m)$ *time.*

1.2 Organization

In Section 2 we provide definitions and some useful known results. In Section 3 we present Merlin-Arthur protocols for the k-SUM problem and several related problems. In Section 4 we present a Merlin-Arthur protocol for counting zero-weight k-cliques, and show that it implies near-optimal protocols for many related fine-grained problems such as APSP. In Section 5 we present a Merlin-Arthur protocol for certifying unsatisfiability of k-CNFs. Then, in Section 6 we describe two barriers for obtaining better Merlin-Arthur protocols. In Section 7 we present a Merlin-Arthur protocol for the True Quantified Boolean Formulas problem. Finally we conclude with some open questions in Section 8.

2 Preliminaries

We assume basic familiarity with computational complexity and algorithms. The following notions will be particularly important for this paper.

Merlin-Arthur Protocols. We say that a function $f : \{0,1\}^\star \to \{0,1\}$ has a **Merlin-Arthur protocol** (or proof system) running in $T(n)$ time with proofs of length $P(n)$ if there is a probabilistic algorithm V such that for all binary strings x with $|x| = n$:

- If $f(x) = 1$, then there is a $y \in \{0,1\}^{P(n)}$ such that $V(x, y)$ accepts in $T(n)$ time, with probability 1.
- If $f(x) = 0$, then for every $y \in \{0,1\}^{P(n)}$, $V(x, y)$ rejects in $T(n)$ time, with probability at least $2/3$.

Concretely, we assume Arthur's verification algorithm V runs in the word-RAM model with words of length $\log(n)$. We only consider protocols where the proof length $P(n)$ is bounded above by the verification time $T(n)$. We often refer to $T(n)$ as the *Merlin-Arthur time* of the protocol. If we say a problem can be "solved in Merlin-Arthur time $T(n)$," we mean it has a Merlin-Arthur protocol running in time $T(n)$.

Williams' Multipoint Evaluation Protocol. We will use Williams' protocol [Wil16] for the Multipoint Circuit Evaluation problem, defined as follows.

Definition 2.1 ([Wil16, Definition 1.1])**.** *The Multipoint Circuit Evaluation problem: given an arithmetic circuit C on n variables over a finite field $\mathbb{F}$, and a list $a_1, \ldots, a_K \in \mathbb{F}^n$, output $(C(a_1), \ldots, C(a_K)) \in \mathbb{F}^K$.*

Theorem 2.2 (Williams [Wil16, Theorem 3.1])**.** *For every prime power q and $\varepsilon > 0$, Multipoint Circuit Evaluation for K points in $(\mathbb{F}_q)^n$ on an arithmetic circuit C of n inputs, s gates, and degree d has an MA-proof system where:*

- *Merlin sends a proof of $O(Kd \cdot \log(Kqd/\varepsilon))$ bits, and,*
- *Arthur tosses at most $\log(Kqd/\varepsilon)$ coins, outputs $(C(a_1), \ldots, C(a_K))$ incorrectly with probability at most ε, and runs in time $(K \cdot \max\{d, n\} + s \cdot \mathrm{poly}(\log s)) \cdot \mathrm{poly}(\log(Kqd/\varepsilon))$.*

Fast Polynomial Evaluation and Interpolation. We need the following classical results on algebraic algorithms. We write $[n] = \{1, 2, \ldots, n\}$ to denote the set of the first n positive integers.

Theorem 2.3 (Fast Multipoint Evaluation [Fid72], multivariate version)**.** *Let k be a fixed positive integer. Given a k-variate polynomial $p(x_1, x_2, \ldots, x_k) \in \mathbb{F}[X_1, \ldots, X_k]$ with each variable having individual degree less than n, presented as at most n^k coefficients, and given kn points $\alpha_{j,i} \in \mathbb{F}$ $(1 \le j \le k, 1 \le i \le n)$, we can compute $p(\alpha_{1,i_1}, \alpha_{2,i_2}, \ldots, \alpha_{k,i_k})$ for all $(i_1, \ldots, i_k) \in [n]^k$ in $\tilde{O}(n^k)$ additions and multiplications in $\mathbb{F}$.*

Theorem 2.4 (Fast Interpolation [Hor72], multivariate version)**.** *Let k be a fixed positive integer. Given kn points $\alpha_{j,i} \in \mathbb{F}$ $(1 \le j \le k, 1 \le i \le n)$, together with the values of $p(\alpha_{1,i_1}, \alpha_{2,i_2}, \ldots, \alpha_{k,i_k}) \in \mathbb{F}$ for all $(i_1, \ldots, i_k) \in [n]^k$, we can output the coefficients of the unique such polynomial $p \in \mathbb{F}[X_1, \ldots, X_k]$ in which every variable has individual degree less than n, in $\tilde{O}(n^k)$ additions and multiplications in $\mathbb{F}$.*

The original references for these two theorems only proved the univariate case ($k = 1$), but one can easily prove the multivariate versions above by applying the univariate algorithms to each variable one by one. Here we provide a sketch of the reduction from multivariate interpolation to univariate interpolation.

The original univariate versions of these theorems were also used in Williams' protocol [Wil16].

Proof Sketch of Theorem 2.4. We will show how the univariate version [Hor72] of the interpolation algorithm easily implies the k-variate case.

We prove the result by induction on k. Recall that the n points on the x_k coordinate are $\alpha_{k,1}, \alpha_{k,2}, \ldots, \alpha_{k,n}$. Consider the $(k-1)$-variate polynomials

$$q_j(x_1, \ldots, x_{k-1}) := p(x_1, \ldots, x_{k-1}, \alpha_{k,j})$$

for all $1 \le j \le n$, obtained from substituting $x_k = \alpha_{k,j}$ into the original k-variate polynomial p. Since for each $1 \le j \le n$ we know the values of $q_j(\alpha_{1,i_1}, \ldots, \alpha_{k-1,i_{k-1}})$ on all $(i_1, \ldots, i_{k-1}) \in [n]^{k-1}$), by the induction hypothesis we know that the $\{q_j\}_{j \in [n]}$ are uniquely determined and can be computed by running the $(k-1)$-variate interpolation algorithm in $\tilde{O}(n \cdot n^{k-1}) = \tilde{O}(n^k)$ total time. Finally, for each tuple

$$(d_1, \ldots, d_{k-1}) \in \{0, 1, \ldots, n-1\}^{k-1}$$

of degrees, the coefficients of the monomials $x_1^{d_1} \cdots x_{k-1}^{d_{k-1}}$ in the polynomials $\{q_j\}_{j \in [n]}$ taken together uniquely determine the coefficients of $x_1^{d_1} \cdots x_{k-1}^{d_{k-1}} x_k^d$ in the polynomial p for all $0 \le d \le n-1$. These coefficients can again be recovered by the univariate interpolation algorithm, taking in $\tilde{O}(n)$ time for each tuple. □

3 An Improved Merlin-Arthur Protocol for k-SUM

Let k be a positive integer. In the k-SUM problem, we are given n integers $a_1, \ldots, a_n$ with magnitude at most n^c for some constant c, and are tasked with determining if there exist indices $i_1, \ldots, i_k$ (not necessarily distinct) such that

$$a_{i_1} + \cdots + a_{i_k} = 0.$$

We call a list of indices $i_1, \ldots, i_k$ satisfying the above equation a *k-SUM solution*. We remark that another popular version of k-SUM from the literature, which we call k-SUM-Distinct, additionally requires the indices $i_1, \ldots, i_k$ in the solution to be distinct. Here, for convenience and consistency with the definition used in [CGI+16], we focus on the k-SUM problem, and later in Section 3.1 note how k-SUM-Distinct can be easily reduced to k-SUM.

In the Merlin-Arthur setting, it is trivial to verify a k-SUM solution exists since Merlin can just send Arthur a solution. Certifying that no k-SUM solutions exist is much more challenging. Our protocol for this problem is based on the following protocol for quickly computing a coefficient in a product of polynomials.

Lemma 3.1. *Let $F_1(x), \ldots, F_k(x)$ be univariate polynomials over $\mathbb{F}_q$ each of degree at most d, for some prime q. Let M be the total number of nonzero coefficients appearing among these polynomials. Then given any integer t and error rate $\delta \in (0,1)$, there is a Merlin-Arthur protocol for determining the coefficient of x^t in the product*

$$P(x) = \prod_{i=1}^{k} F_i(x)$$

with proof size $\tilde{O}\left((k\sqrt{d})(\log q)\right)$ and runtime $\tilde{O}\left((M + k\sqrt{d})(\log q)(\log(1/\delta))\right)$.

Proof. We may assume that $0 \le t \le kd$, since otherwise the coefficient of x^t in P is zero. Let

$$P(x) = \sum_{\ell=0}^{kd} p_\ell x^\ell$$

be the product of all the F_i polynomials.

Set $m = \lfloor \sqrt{d} \rfloor$. The protocol works as follows.

1. For each nonnegative integer $\ell \le kd$ with $\ell \equiv t \pmod m$, Merlin sends Arthur some $c_\ell \in \mathbb{F}_q$. Each such c_ℓ term is Merlin's claim for the value of the corresponding coefficient p_ℓ in $P(x)$.

2. Arthur takes an integer h such that $q^h \ge kd/\delta$ and then samples $w \in \mathbb{F}_{q^h}$ uniformly at random. To construct the field $\mathbb{F}_{q^h}$, Arthur just needs a polynomial of degree h irreducible over $\mathbb{F}_q$. As noted in [Wil16], we can do this efficiently by having Merlin send such a polynomial, and then having Arthur verify the polynomial is irreducible in asymptotically

$$h^{1+o(1)}(\log q)^{2+o(1)} = \left(\left(\log(kd) + \log(1/\delta)\right)(\log q)\right)^{1+o(1)}$$

time using known irreducibility tests [KU11, Section 8.2].

For the rest of this protocol, Arthur performs all computations over $\mathbb{F}_{q^h}$.

For each polynomial

$$F_i(x) = \sum_{\ell=0}^{d} f_{i,\ell} x^\ell$$

of degree at most d, we say its *reduced form* is the polynomial

$$G_i(x) = \sum_{b=0}^{m-1} \left(\sum_{\ell \equiv b \bmod m} w^\ell f_{i,\ell} \right) x^b, \tag{1}$$

formed by reducing the polynomial $F_i(wx)$ modulo $x^m - 1$.

Arthur first constructs the reduced forms G_i of F_i for each $1 \le i \le k$, in $\tilde{O}(M)$ time. Then, using fast polynomial multiplication, Arthur computes the product

$$R(x) = \prod_{i=1}^{k} G_i(x) = \sum_{\ell=0}^{k(m-1)} r_\ell x^\ell$$

in $\tilde{O}(km)$ time. By adding the coefficients of this polynomial and appealing to the definition of the reduced polynomials in Eq. (1), Arthur can compute the quantity

$$\sum_{\ell \equiv t \bmod m} r_\ell = \sum_{\substack{b_1+\cdots+b_k \equiv t \bmod m \\ \ell_i \equiv b_i \bmod m\ \forall i \in [k]}} w^{\ell_1+\cdots+\ell_k} \prod_{i=1}^{k} f_{i,\ell_i} = \sum_{\substack{\ell \equiv t \bmod m \\ \ell_1+\cdots+\ell_k=\ell}} w^\ell \prod_{i=1}^{k} f_{i,\ell_i} = \sum_{\ell \equiv t \bmod m} p_\ell w^\ell. \tag{2}$$

In the second summation above, we are summing over a subset of k-tuples $(\ell_1, \ldots, \ell_k)$ with the property that $0 \le \ell_i \le d$ for each i. We define b_i to be the residue of ℓ_i modulo m for each i, and only consider those k-tuples in the sum if the sum of their residues modulo m is congruent to t modulo m. In the transition from the second to the third summation above, we note that this is equivalent to summing over all k-tuples $(\ell_1, \ldots, \ell_k)$ such that $0 \le \ell_i \le d$ for each i and the sum of the ℓ_i is congruent to some integer t modulo m.

After computing the sum from Equation (2), Arthur also computes

$$\sum_{\ell \equiv t \bmod m} c_\ell w^\ell \tag{3}$$

in $\tilde{O}(M)$ time using the values Merlin sent. If this sum and the value of the sum from Eq. (2) agree over $\mathbb{F}_{q^h}$, then Arthur accepts and returns c_t as the coefficient of x^t in $P(x)$. If the sums disagree, then Arthur rejects the proof.

In the above protocol, if Merlin sends integers with $c_\ell = p_\ell$ for all $\ell \le kd$ with $\ell \equiv t \pmod m$, then the values from Eq. (2) and Eq. (3) will agree. If this happens, Arthur will accept and correctly determine p_t as the value of the desired coefficient.

The only way for Arthur to accept an incorrect value for the coefficient is if $q_t \neq p_t$. In this case,

$$Q(x) = \sum_{\ell \equiv t \bmod m} p_\ell x^\ell$$

and

$$C(x) = \sum_{\ell \equiv t \bmod m} c_\ell x^\ell$$

are distinct polynomials over $\mathbb{F}_q$ of degree at most kd. This means they agree on at most kd points. So, for uniform random $w \in \mathbb{F}_{q^h}$, we have $Q(w) \neq C(w)$ with probability at least

$$1 - \frac{kd}{q^h} \ge 1 - \delta$$

by our choice of h. Thus with probability at least $1-\delta$, Arthur rejects an incorrect proof. □

Remark 3.2. Recall that in the Subset Sum problem, we are given input integers $a_1, \ldots, a_n$, and must decide if some collection of the inputs sums to a given target integer t. Although framed somewhat differently, Nederlof's Merlin-Arthur protocol for Subset Sum from [Ned17] employs a similar tactic, and can be recovered by applying Lemma 3.1 to check if the coefficient of x^t in the product

$$(1+x^{a_1})(1+x^{a_2})\cdots(1+x^{a_n})$$

is nonzero or not.

Reminder of Theorem 1.1. *For any fixed integer $k \ge 3$, certifying that a list of n integers has no k-SUM solution can be done in Merlin-Arthur time $\tilde{O}(n^{k/3})$.*

Before proving Theorem 1.1, we first informally describe the three primary ideas underlying our Merlin-Arthur protocol.

First, solving k-SUM corresponds to checking the coefficient of some product of polynomials, where the degree of the polynomials is related to the magnitude $\max_i |a_i|$ of the input integers. This is hard in general since these magnitudes could be large polynomials in n, but could be made more efficient if there was a simple way to reduce the sizes of the inputs.

The second idea comes from the conondeterministic algorithm of [CGI+16, Lemma 5.8] for k-SUM: we have Merlin send a small prime p such that "few" sums of the k input integers vanish modulo p. Given p, Arthur can easily count the number of these sums (intuitively, because Arthur can replace each a_i with its residue modulo p to reduce the size of the input integers). If Merlin then sends all the k-tuples of inputs that sum to zero modulo p, Arthur can check that the number of tuples sent matches the count computed, and then scan through the list to verify that none of the given sums equal zero over the integers.

The third and final idea is to employ the protocol for fast polynomial multiplication from Lemma 3.1.

We now describe the protocol.

Proof of Theorem 1.1. Suppose we are given a k-SUM instance on n integers $a_1, \dots, a_n \in [-n^c, n^c]$ for some constant $c > 0$. Merlin first sends a prime $p = \tilde{\Theta}(n^{2k/3})$. Let

$$S = \{(i_1, \dots, i_k) \mid a_{i_1} + \cdots + a_{i_k} \equiv 0 \pmod{p}\}$$

be the the set of k-tuples whose sums vanish modulo p. Merlin additionally sends a set T of k-tuples of indices such that $|T| \le \tilde{O}(n^{k/3})$ and claims that $T = S$.

Now, for each $i \in [n]$ let b_i be the residue of a_i modulo p. Define the polynomial

$$B(x) = \sum_{i=1}^{n} x^{b_i}.$$

The coefficients of the k^{th} power of this polynomial encode information that will help us solve the k-SUM problem. In particular, we leverage the following simple observation.

Claim 3.3. *The sum of the coefficients of $x^0, x^p, \dots,$ and $x^{(k-1)p}$ of the polynomial*

$$B(x)^k$$

is equal to the number of k-tuples $(i_1, \dots, i_k) \in [n]^k$ such that

$$a_{i_1} + \cdots + a_{i_k} \equiv 0 \pmod{p}.$$

Proof. We can expand

$$B(x)^k = \left(\sum_{i=1}^{n} x^{b_i}\right)^k = \sum_{i_1, \dots, i_k} x^{b_{i_1} + \cdots + b_{i_k}} = \sum_{\ell=0}^{k(p-1)} s_\ell x^\ell$$

where s_ℓ denotes the number of k-tuples $(i_1, \dots, i_k) \in [n]^k$ such that

$$b_{i_1} + \cdots + b_{i_k} = \ell.$$

Now, since each b_i is the residue of a_i modulo p, we know that

$$a_{i_1} + \cdots + a_{i_k} \equiv 0 \pmod{p}$$

precisely when

$$b_{i_1} + \cdots + b_{i_k} \equiv 0 \pmod{p}.$$

Because $0 \le b_i \le p-1$ for each index i, we know that

$$b_{i_1} + \cdots + b_{i_k} \equiv 0 \pmod{p}$$

if and only if

$$b_{i_1} + \cdots + b_{i_k} \in \{0, p, \dots, (k-1)p\}.$$

Combining these observations, we get that

$$s_0 + s_p + \cdots + s_{(k-1)p}$$

is equal to the number of k-tuples $(i_1, \ldots, i_k) \in [n]^k$ such that

$$a_{i_1} + \cdots + a_{i_k} \equiv 0 \pmod p$$

which proves the desired result. □

Returning to the protocol, Merlin and Arthur run k instances of the protocol from Lemma 3.1 in parallel[8], for a field of size q, for some prime $q > n^k$, with error rate $\delta = 1/(kn)$, to determine the coefficients $s_{p\ell}$ of $x^{p\ell}$ in $B(x)^k$ for all $\ell \in \{0, 1, \ldots, k-1\}$. Arthur rejects if Merlin fails to convince him of the values of any of these coefficients.

Otherwise, Arthur checks that

$$s_0 + s_p + \cdots + s_{(k-1)p} = |T|. \tag{4}$$

He also checks that for each $(i_1, \ldots, i_k) \in T$, we have

$$a_{i_1} + \cdots + a_{i_k} \neq 0$$

over the integers. If both these checks pass, Arthur accepts. Otherwise, he rejects.

We now explain why this Merlin-Arthur proof system is correct.

First, suppose that no k of the a_i sum to zero. We show that Merlin has a proof which always convinces Arthur to accept.

By the prime number theorem, there exists some constant C such that there are at least $n^{2k/3}$ distinct primes in the interval $I = [n^{2k/3}, Cn^{2k/3}\log n]$. Now, by assumption, each sum

$$a_{i_1} + \cdots + a_{i_k} \tag{5}$$

is a nonzero integer with magnitude at most kn^c, and thus has at most $c(\log n + \log k)$ distinct prime divisors. Thus by the pigeonhole principle, there exists a prime in the interval I which divides at most

$$\frac{n^k \cdot c(\log n + \log k)}{n^{2k/3}} \leq \tilde{O}(n^{k/3})$$

of the n^k sums of the form presented in Eq. (5).

So, Merlin can send a prime p satisfying the desired properties to Arthur. He also sends $T = S$, the list of sums of the form given in Eq. (5) which are divisible by p, which has $|T| \leq \tilde{O}(n^{k/3})$ by the choice of p. If Merlin sends the correct values for $s_0, s_p, \ldots, s_{(k-1)p}$, then Eq. (4) will hold by Claim 3.3, and Arthur accepts. Now, suppose that some k of the a_i do in fact sum to zero. In this case, we show that with high probability Arthur will reject.

First, if the set T which Merlin sends contains a k-tuple corresponding to a list of k inputs whose sum does not vanish modulo p, or a sum which sums to zero over the integers, then Arthur will automatically reject. Otherwise, by assumption, the set T is missing some tuple $(j_1, \ldots, j_k)$ such that

$$a_{j_1} + \cdots + a_{j_k} = 0.$$

Reducing the above equation modulo p, we see that

$$b_{j_1} + \cdots + b_{j_k} \equiv 0 \pmod p.$$

Then by Claim 3.3, if Merlin and Arthur have decided on the correct coefficients of $B(x)^k$, we have

$$s_0 + s_p + \cdots + s_{(k-1)p} < |T|$$

and Arthur will reject. By union bound and Lemma 3.1, Arthur correctly rejects with probability at least

$$1 - k\delta = 1 - \frac{1}{n}.$$

□

[8] By running k protocols in parallel, we mean that Merlin concatenates the k messages he would send Arthur in each protocol, and Arthur verifies each message independently.

3.1 Implications of the k-SUM protocol

We now show the implications of our k-SUM protocol.

We first consider the k-SUM-Partitioned problem, where we are given k input lists $A^{(1)}, A^{(2)}, \dots, A^{(k)}$ each consisting of n integers from $[-n^c, n^c]$ for some constant c, and want to determine if there exist indices $i_1, \dots, i_k$ such that $A^{(1)}_{i_1} + A^{(2)}_{i_2} + \cdots + A^{(k)}_{i_k} = 0$ (this problem has also been called k-SUM' [GO95] and Colorful k-SUM). We note there is a deterministic reduction from k-SUM-Partitioned to k-SUM, extending the case of 3-SUM [GO95].

Corollary 3.4. *For any fixed integer $k \geq 3$, certifying that a k-SUM-Partitioned instance has no solution can be done in Merlin-Arthur time $\tilde{O}(n^{k/3})$.*

Proof Sketch. Let $M = \lceil 10kn^c \rceil$. We create a k-SUM instance as follows. For every $1 \leq i \leq k$ and every integer a from the input list $A^{(i)}$, we include the integer

$$\begin{cases} 2^i \cdot M + a & \text{if } 1 \leq i \leq k-1, \\ -(2 + 4 + \cdots + 2^{k-1}) \cdot M + a & \text{if } i = k \end{cases}$$

in the k-SUM instance. A solution to the k-SUM-Partitioned immediately implies a solution of the new k-SUM instance. Conversely, it can be shown that any k-SUM solution must recover a solution of the k-SUM-Partitioned instance.

Hence, applying the protocol from Theorem 1.1 to this k-SUM instance of kn integers can solve the original k-SUM-Partitioned instance. □

Recall that in the k-SUM-Distinct problem, we are given n integers $a_1, \dots, a_n$ with magnitude at most n^c and need to determine if there exist k *distinct* indices $i_1, \dots, i_k$ such that $a_{i_1} + \cdots + a_{i_k} = 0$. We will use a folklore deterministic reduction from k-SUM-Distinct to k-SUM-Partitioned.

Corollary 3.5. *For any fixed integer $k \geq 3$, certifying that a list of n integers has no k-SUM-Distinct solution can be done in Merlin-Arthur time $\tilde{O}(n^{k/3})$.*

Proof. Given a k-SUM-Distinct instance $a_1, a_2, \dots, a_n$, we will deterministically create $(\log n)^{O(1)}$ many instances of k-SUM-Partitioned in which the k input lists are disjoint subsets of $\{a_1, \dots, a_n\}$, so that every possible k distinct indices $i_1, \dots, i_k \in [n]$ are isolated at least once. Then the k-SUM-Distinct instance has no solution if and only if none of these k-SUM-Partitioned instances have solutions, and the statement then follows from Corollary 3.4.

Take any k distinct indices $i_1, \dots, i_k \in [n]$ and consider their binary expansions

$$\mathsf{bin}(i_1), \dots, \mathsf{bin}(i_k) \in \{0,1\}^{\log n}.$$

Observe that there must exist a set of coordinates $C \subseteq [\log n]$ of size $|C| \leq k-1$, so that the projections $\mathsf{bin}(i_1)|_C, \dots, \mathsf{bin}(i_k)|_C$ are still distinct. Hence, for every possibility of

$$(\mathsf{bin}(i_1)|_C, \dots, \mathsf{bin}(i_k)|_C) = (v_1, v_2, \dots, v_k) \in \left(\{0,1\}^{k-1}\right)^k,$$

we create a k-SUM-Partitioned instance $(A^{(1)}, \dots, A^{(k)})$ where $A^{(j)} = \{a_i : i \in [n], \mathsf{bin}(i)|_C = v_j\}$. The total number of instances created is at most $c_k \cdot (\log n)^{k-1}$ for some constant c_k. □

By another folklore reduction, our protocol for the 3-SUM-Partitioned problem immediately implies an improved protocol for the Subset Sum problem.

Reminder of Corollary 1.2. *Certifying that a list of n integers has no Subset Sum solution can be done in $2^{n/3} \cdot \mathrm{poly}(n)$ Merlin-Arthur time.*

Proof. Suppose we have an instance of Subset Sum consisting of n inputs $a_1, a_2, \dots, a_n$ and a target integer t. We partition the set $[n]$ into the disjoint union of three subsets $A, B, C \subseteq [n]$, each with size at most $\lceil n/3 \rceil$, and define the sets

$$X = \left\{ \sum_{i \in S} a_i \mid S \subseteq A \right\}, \quad Y = \left\{ \sum_{i \in S} a_i : S \subseteq B \right\}, \quad Z = \left\{ -t + \sum_{i \in S} a_i : S \subseteq C \right\}.$$

Then there exists a subset $S \subseteq [n]$ such that $\sum_{i \in S} a_i = t$ if and only if there exist $x \in X, y \in Y, z \in Z$ such that $x + y + z = 0$, which is a 3-SUM-Partitioned instance. Note that X, Y, Z each have at most $2^{\lceil n/3 \rceil}$ elements. Applying Corollary 3.4 solves the problem in $2^{n/3} \cdot \mathrm{poly}(n)$ time. □

Reminder of Corollary 1.4. *For any fixed integer $k \geq 3$, the All-Numbers k-SUM problem can be solved in Merlin-Arthur time $\tilde{O}(n^{k/3})$.*

Proof. For every index i such that a_i is part of a k-SUM solution, Merlin simply sends a witnessing solution to Arthur. Let $S \subseteq [n]$ be the set of remaining indices, which do not participate in any solution. It remains to verify that S is correct.

Denoting $M := \max_{i \in [n]} |a_i|$, construct a k-SUM instance with input integers $A \cup B$, where

$$A := \{10M + a_i : i \in [n]\} \quad \text{and} \quad B := \{-10(k-1)M + a_i : i \in S\}.$$

By our choice of M, every k-SUM solution in this new instance must use exactly one integer from B, and hence corresponds to a k-SUM solution in the original instance that uses a_i for some $i \in S$. So it suffices to use the protocol from Theorem 1.1 to prove that this new k-SUM instance has no solution. □

Reminder of Corollary 1.3. *MinPlus Convolution can be solved in Merlin-Arthur time $\tilde{O}(n)$.*

Proof Sketch. Merlin first sends the correct values of c_k, each accompanied with a witness pair (i, j) such that $i + j = k$ and $a_i + b_j = c_k$. Then it remains to verify that $a_i + b_j \geq c_{i+j}$ for all i, j, which is equivalent to the MaxConv UpperBound problem defined in [CMWW19].

In [CMWW19, Appendix A], it was shown (using the techniques from [VW13, Theorem 3.3]) that MaxConv UpperBound can be deterministically reduced to the 3-SUM Convolution problem. In this problem, we are given three integer arrays a, b, c and want to decide whether there exists a pair of indices (i, j) such that

$$a_i + b_j = c_{i+j}.$$

The 3-SUM Convolution problem easily reduces to the 3-SUM-Partitioned problem on lists $\{a'_i\}, \{b'_j\}, \{c'_k\}$ defined as $a'_i = iM + a_i, b'_j = jM + b_j, c'_k = -kM - c_k$ for large enough M, which can be solved by the near-linear-time Merlin Arthur protocol from Corollary 3.4. □

In the Zero-Weight Triangle problem, we are given an undirected graph G on n vertices and m edges with weights from $[-n^c, n^c]$ for some positive constant c, and are tasked with determining if G contains a triangle whose edge weights sum to zero.

Corollary 3.6. *Certifying that a given graph has no zero-weight triangles can be done in Merlin-Arthur time $\tilde{O}(m)$.*

Proof Sketch. We first make the graph directed by replacing each edge connecting vertices u and v with two arcs, one going from u to v and the other going from v to u. Then by making three copies of the original graph, we may assume without loss of generality that the graph is tripartite with three parts A, B, C, and edges are oriented from A to B, B to C, and C to A.

We use the reduction described in [JV16]. Merlin first assigns integer node labels $0 \leq \ell(u) \leq \mathrm{poly}(n)$ to each node u in the graph. For each edge (u, v) of weight w, insert an integer $\ell(u) - \ell(v) + w$ to the 3SUM instance. Then it is easy to see that any zero-weight triangle (a, b, c) would lead to a 3SUM solution

$$(\ell(a) - \ell(b) + w(a, b)) + (\ell(b) - \ell(c) + w(b, c)) + (\ell(c) - \ell(a) + w(c, a)) = 0.$$

On the other hand, if the graph does not contain a zero-weight triangle, then a simple probabilistic argument implies the existence of a way to pick the node labels so that the resulting 3-SUM instance has no solution. □

In the next section we will see that we can actually *count* the number of zero-weight triangles by a different Merlin-Arthur protocol with essentially the same time complexity.

4 Counting Zero-Weight Cliques

In this section we present the Merlin-Arthur protocol for #Zero-Weight k-Clique and prove Theorem 1.5, which is restated below. We assume the input graph is a simple undirected graph with n nodes and m weighted edges, where $m \ge \Omega(n)$ and all edge weights are in $[-n^c, n^c]$ for some constant c.

Reminder of Theorem 1.5. *For any fixed integer $k \ge 3$, #Zero-Weight k-Clique can be solved by a Merlin-Arthur protocol with proof length $\tilde{O}(n^{\lfloor k/2 \rfloor})$ and verification time $\tilde{O}\big(n^{\lceil k/2 \rceil} \cdot (m/n^2)^{\lfloor (k+1)/4 \rfloor} + n^{\lfloor k/2 \rfloor}\big)$.*

Proof. Without loss of generality, we assume that the input graph is a k-partite graph with k parts of nodes $A_1, A_2, \dots, A_{\lfloor k/2 \rfloor}, B_1, B_2, \dots, B_{\lceil k/2 \rceil}$, each containing n nodes, and every edge connects two nodes coming from different parts. We identify the nodes with integers $\{1, 2, \dots, kn\}$. We also denote

$$A := \bigcup_{i=1}^{\lfloor k/2 \rfloor} A_i \quad \text{and} \quad B := \bigcup_{j=1}^{\lceil k/2 \rceil} B_j.$$

We encode the edge weights in binary, and will use arrow notation to emphasize that they are bit-vectors of length $O(\log n)$. For each node $b \in B$ and node $a \in A$, let $\vec{f_b}(a)$ be the binary encoding of the weight of edge (a, b) (if this edge does not appear in the input graph, we simply treat its edge weight as a large enough positive number M so that it can never participate in a zero-weight k-clique). We extend $\vec{f_b}(a)$ to a vector polynomial $\vec{f_b}(x)$ of degree $|A| = O(n)$. Note that $\vec{f_b}(x)$ consists of $O(\log n)$ many scalar polynomials each corresponding to one bit in the binary encoding of edge weights. These scalar polynomials are over the field $\mathbb{F}_p$ for some prime $p = \mathrm{poly}(n)$ and $p > n^k$.

Then, define a $\lfloor k/2 \rfloor$-variate vector polynomial $\vec{h}(x_1, \dots, x_{\lfloor k/2 \rfloor})$, such that for every

$$a_1 \in A_1, \dots, a_{\lfloor k/2 \rfloor} \in A_{\lfloor k/2 \rfloor},$$

the vector $\vec{h}(a_1, \dots, a_{\lfloor k/2 \rfloor})$ encodes the total weight of the clique formed by the nodes $a_1, \dots, a_{\lfloor k/2 \rfloor}$. Note that $\vec{h}(x_1, \dots, x_{\lfloor k/2 \rfloor})$ has individual degree $O(n)$.

For nodes $b_1 \in B_1, \dots, b_{\lceil k/2 \rceil} \in B_{\lceil k/2 \rceil}$, let $\vec{w}(b_1, \dots, b_{\lceil k/2 \rceil})$ denote the binary encoding of the total weight of the clique formed by nodes $b_1, \dots, b_{\lceil k/2 \rceil}$. Then, define

$$\begin{aligned} &P(x_1, \dots, x_{\lfloor k/2 \rfloor}) \\ := &\sum_{\substack{b_1 \in B_1, \dots, b_{\lceil k/2 \rceil} \in B_{\lceil k/2 \rceil} \\ \text{forming a clique}}} Q\big(\vec{h}(x_1, \dots, x_{\lfloor k/2 \rfloor}),\ \vec{w}(b_1, \dots, b_{\lceil k/2 \rceil}),\ \vec{f}_{b_1}(x_1), \vec{f}_{b_1}(x_2), \dots, \vec{f}_{b_{\lceil k/2 \rceil}}(x_{\lfloor k/2 \rfloor})\big), \end{aligned} \tag{6}$$

where Q takes $2 + \lceil k/2 \rceil \cdot \lfloor k/2 \rfloor$ input integers (encoded in binary), and outputs 1 if the input integers sum to exactly zero, and outputs 0 otherwise. Hence, by definition,

$$\sum_{a_1 \in A_1, \dots a_{\lfloor k/2 \rfloor} \in A_{\lfloor k/2 \rfloor}} P(a_1, \dots, a_{\lfloor k/2 \rfloor}) \tag{7}$$

equals the number of zero-weight k-cliques in the input graph.

Note that Q only involves a constant number of additions and a comparison, which can be implemented by an AC^0 circuit with $O(\log n)$ input gates and $\mathrm{polylog}(n)$ size. We can convert Q into an equivalent arithmetic circuit of $\mathrm{polylog}(n)$ size and degree. It then follows that P is a polynomial (over $\mathbb{F}_p$) of degree at most $n \cdot \mathrm{polylog}(n)$.

At the beginning of the protocol, Merlin sends the polynomial P defined in Equation (6) to Arthur, represented as $\tilde{O}(n^{\lfloor k/2 \rfloor})$ many coefficients. Then, Arthur can evaluate the values of $P(a_1, \dots, a_{\lfloor k/2 \rfloor})$ for all $(a_1, \dots, a_{\lfloor k/2 \rfloor}) \in A_1 \times \cdots \times A_{\lfloor k/2 \rfloor}$ in $\tilde{O}(n^{\lfloor k/2 \rfloor})$ time using Theorem 2.3. Then Arthur can easily compute the count of zero-weight k-cliques using Equation (7).

To verify that P is correct, Arthur picks random $r_1, \dots, r_{\lfloor k/2 \rfloor} \in \mathbb{F}_p$ and verifies that Equation (6) holds at the point $(x_1, \dots, x_{\lfloor k/2 \rfloor}) = (r_1, \dots, r_{\lfloor k/2 \rfloor})$. In order to evaluate the sum in Equation (6) (with $x_1 := r_1, \dots, x_{\lfloor k/2 \rfloor} := r_{\lfloor k/2 \rfloor}$), Arthur first needs to perform the following preprocessing steps by interpolation:

1. Compute $\vec{h}(r_1, \ldots, r_{\lfloor k/2 \rfloor})$.

2. For every node $b \in B$, compute $\vec{f_b}(r_1), \ldots, \vec{f_b}(r_{\lfloor k/2 \rfloor})$.

After performing these preprocessing steps, Arthur can straightforwardly evaluate Equation (6) at the chosen point, which involves at most $m^{\lfloor \lceil k/2 \rceil / 2 \rfloor} \cdot n^{\lceil k/2 \rceil \bmod 2} = n^{\lceil k/2 \rceil} \cdot \big(m/n^2\big)^{\lfloor (k+1)/4 \rfloor}$ summands (since $(b_1, b_2), (b_3, b_4), \ldots$ must be edges in the input graph).

It remains to analyze the time complexity for these preprocessing steps.

- Step 1. Compute $\vec{h}(r_1, \ldots, r_{\lfloor k/2 \rfloor})$.

  This can be done by straightforward interpolation in $\tilde{O}(n^{\lfloor k/2 \rfloor})$ time (Theorem 2.4).

- Step 2. For every node $b \in B$, compute $\vec{f_b}(r_1), \ldots, \vec{f_b}(r_{\lfloor k/2 \rfloor})$.

  Let $N(b)$ denote the set of neighbors of node b in A. We will show that, after $\tilde{O}(n)$-time preprocessing, this step can be performed in $|N(b)| \cdot \operatorname{polylog}(n)$ time for every node b, and thus the total running time is $\tilde{O}(n + \sum_{b \in B} |N(b)|) = \tilde{O}(m)$.

  Recall that $\vec{f_b}(a)$ encodes the edge weight of $w(a, b)$ if $a \in N(b)$, or encodes integer M if $a \notin N(b)$. Since the $O(\log n)$ coordinates of the vector will be considered separately, in the following we only need to discuss how to process one of these coordinates. Abusing notation, we use $f_b(a)$ to indicate the value of $\vec{f_b}(a)$ on the coordinate under consideration, and use $w(a, b)$ and M to denote the corresponding values on this coordinate. That is, $f_b(a) = w(a, b)$ if $a \in N(b)$, and $f_b(a) = M$ if $a \in A \setminus N(b)$.

  By Lagrange interpolation, we have

  $$f_b(x) = M + \sum_{a \in N(b)} (w(a, b) - M) \cdot \frac{\prod_{a' \in A \setminus \{a\}} (x - a')}{\prod_{a' \in A \setminus \{a\}} (a - a')}.$$

  The denominator $\prod_{a' \in A \setminus \{a\}} (a - a')$ can be easily computed for all a in $\tilde{O}(n)$ total time (recall that the node set A is identified with the integer set $\{1, 2, \ldots, \lfloor k/2 \rfloor \cdot n\}$). For each $x = r_i$, one can perform a simple $\tilde{O}(n)$-time preprocessing so that for each $a \in A$, the numerator $\prod_{a' \in A \setminus \{a\}} (r_i - a')$ can be computed in constant field operations. Then, it only takes $O(|N(b)|)$ field operations to evaluate $f_b(r_i)$.

In summary, the total time complexity for Arthur is $\tilde{O}\big(n^{\lceil k/2 \rceil} \cdot \big(m/n^2\big)^{\lfloor (k+1)/4 \rfloor} + n^{\lfloor k/2 \rfloor} + m\big)$, and the proof length is $\tilde{O}(n^{\lfloor k/2 \rfloor})$. □

We remark that the protocol of Theorem 1.5 can be also used to count cliques with other kinds of restrictions on the edge weights, by simply modifying the predicate Q in Equation (6). For example, our protocol can also apply to the #Negative k-Clique problem, which asks to count the number of k-cliques whose sum of edge weights is negative.

Corollary 4.1. *Theorem 1.5 still holds if we replace #Zero-Weight k-Clique by #Negative k-Clique.*

By modifying Q in Eq. (6) we can also count the number of any 4-node (induced or not-necessarily-induced) subgraphs in the input graph, in near-optimal $\tilde{O}(n^2)$ Merlin-Arthur time. See [VWWY15] for the best known algorithms to detect 4-node subgraphs in the input graph.

Corollary 4.2. *For any 4-node pattern graph H, counting the number of (induced or not-necessarily-induced) copies of H in the input graph can be done in $\tilde{O}(n^2)$ Merlin-Arthur time.*

Combining known reductions with Corollary 4.1, our protocol for #Negative Triangle implies near-optimal protocols for MinPlus Product and APSP. Recall that in the MinPlus Product problem, we are given two $n \times n$ integer matrices A, B, and want to compute matrix C defined as $C_{i,j} = \min_{k=1}^{n} \{A_{i,k} + B_{k,j}\}$.

Proof of Corollary 1.6. We first show that MinPlus Product can be solved in Merlin-Arthur time $\tilde{O}(n^2)$. Merlin first sends to Arthur the correct product C, together with the witness $\arg\min_{k=1}^{n}\{A_{i,k}+B_{k,j}\}$ for each entry $C_{i,j}$ in the product. Arthur checks the validity of these witnesses, and then verifies that $A_{i,k}+B_{k,j} \geq C_{i,j}$ hold for all i,j,k. This task easily reduces to the Negative Triangle problem [VW18] as follows: create a tripartite graph (X,Y,Z) with edge weights defined as $w(X_i,Y_k)=A_{i,k}, w(Y_k,Z_j)=B_{k,j}, w(Z_j,X_i)=-C_{i,j}$, and certify that this new graph has no negative triangles, using Corollary 4.1.

Using the Merlin-Arthur protocol for MinPlus Product, we immediately obtain an $\tilde{O}(n^2)$ time Merlin-Arthur protocol for APSP via the standard repeated squaring procedure. In particular, Merlin can send the matrices obtained from all $O(\log n)$ repeated squarings upfront, along with $\tilde{O}(n^2)$-length proofs of their correctness; Arthur can verify each squaring is correct in $\tilde{O}(n^2)$ time, one by one. □

Given a simple undirected graph and a parameter t, the Triangle Listing problem [Pǎt10, BPVZ14, VX20] asks to report $\min(t,z)$ triangles in the graph, where z denotes the total number of triangles in the graph. Our results immediately imply a near-optimal protocol for this task.

Corollary 4.3. *Triangle Listing can be solved in Merlin-Arthur time $\tilde{O}(m+t)$.*

Proof. Merlin uses Theorem 1.5 to prove that the input graph has z triangles in $\tilde{O}(m)$ time, and then sends $\min(t,z)$ many triangles to Arthur, who verifies that these triangles are valid and distinct. □

5 Unsatisfiability of k-CNFs

In this section, we will present a $2^{n-n/O(k)} \cdot \text{poly}(n,m)$ time Merlin-Arthur protocol for k-UNSAT with n variables and m clauses in Theorem 1.7 (note that $m \leq O(n^k)$ in a k-CNF formula). This beats the previously known protocol for k-UNSAT running in $2^{n/2} \cdot \text{poly}(n,m)$ time, which follows directly from [Wil16, Theorem 3.4]. We need the following useful theorems.

Theorem 5.1 (Impagliazzo-Paturi [IP01, Lemma 2])**.** *Let F be a k-CNF formula on m clauses such that every satisfying assignment to F has at least δn variables set to true for any $\delta > 0$. For any $\epsilon > 0$, there exists a $k' > 0$ and F', which is a disjunction of at most $2^{\epsilon n}$ k'-CNFs on at most $n(1-\delta/(ek))$ variables such that F is satisfiable iff F' is satisfiable. Moreover F' can be computed from F in $2^{2\epsilon n}\text{poly}(m)$ time.*

Theorem 5.2 (#SAT for Boolean formulas [Wil16, Theorem 3.4])**.** *For any $k > 0$, #SAT for Boolean formulas with n variables and m connectives has a Merlin-Arthur proof system using $2^{n/2}\text{poly}(n,m)$ time with randomness $O(n)$ and error probability $1/\exp(n)$.*

Recall that the *binary entropy* function $H(\cdot)$ is defined by taking

$$H(p) = -p\log p - (1-p)\log(1-p)$$

for all $p \in (0,1)$. We prove the following result.

Theorem 5.3. *For all $\delta \in (0,1/2)$ and all sufficiently large integers $k > 0$, k-UNSAT has a Merlin-Arthur protocol that runs in time*

$$\left(2^{n(1/2-\delta/(6k))} + 2^{H(\delta)n}\right)\text{poly}(n,m).$$

Proof. The idea behind this protocol is to handle the assignments with fewer than δn variables set to true, and the assignments with more than δn variables set to true, separately. Once we have verified that there are no assignments with δn variables set to true, we can make use of Theorem 5.1 to decompose the formula into formulas with fewer variables. Formally, the protocol proceeds as follows:

Given a k-CNF F on n variables and m clauses, Merlin and Arthur certify the unsatisfiability of F as follows:

1. Arthur enumerates over all possible $O(2^{H(\delta)n})$ assignments with at most δn variables set to true and verifies that none of them satisfy F.

2. Arthur uses Theorem 5.1 with $\epsilon = 1/k^2$ to obtain at most $t = 2^{n/k^2}$ k'-CNFs $F'_1 \dots F'_t$ on $n(1-\delta/(ek))$ variables each.

3. Then, Merlin and Arthur run the protocol from Theorem 5.2, with Merlin sending the proofs for each of $F'_1 \dots F'_t$ and Arthur verifying their unsatisfiability, taking $2^{n(1/2-\delta/(2ek))}\text{poly}(n,m)$ time for each.[9]

Verifying unsatisfiability for all the F'_i's in step 3 takes time

$$2^{n/k^2} \cdot 2^{n/2-\delta/(2ek)}\text{poly}(n,m) \le 2^{n/2-\delta n/(6k)}\text{poly}(n,m),$$

where the inequality holds for sufficiently large k (for example, $k \ge 60$ suffices). Thus, the total time taken by Arthur for verification is $\left(2^{n/2-\delta n/(6k)} + 2^{H(\delta)n}\right)\text{poly}(n,m)$. This completes the proof. □

Reminder of Theorem 1.7. *There is a universal constant $\delta > 0$ such that for all sufficiently large integers $k > 0$, we can verify any unsatisfiable n-variable m-clause k-CNF with a Merlin-Arthur protocol running in $2^{n(1/2-\delta/k)} \cdot \text{poly}(n,m)$ time.*

Proof. We apply Theorem 5.3 by setting $\delta \in (0, 1/2)$ to be small enough that $H(\delta) \le 2\delta \log_2(1/\delta) \le 1/2-\delta/k$ holds for every $k \ge 1$. Then, the protocol of Theorem 5.3 runs in $O(2^{n(1/2-\delta/(6k))})$ time for all large enough integers k. □

6 Faster MA Protocols Require Non-Algebrizing Techniques

In this section, we observe that:

(a) Williams' protocol for #SAT (which runs in $\text{poly}(n,m) \cdot 2^{n/2}$) algebrizes, and

(b) No algebrizing Merlin-Arthur protocol for UNSAT runs in time $2^{n/2}/n^{\omega(1)}$, even for unsatisfiability of 1-CNF formulas.

We stress that both of these are *observations*, which do not require any significant ideas that are not already in the literature. However, we find them striking to consider in the context of our other Merlin-Arthur protocols such as Theorem 1.7, which beat $2^{n/2}$ time by exploiting the structure of k-CNF formulas.

First, we observe that Williams' protocol naturally algebrizes. Let $A : \{0,1\}^\star \to \{0,1\}$ be an arbitrary oracle. For a constant $k \in \mathbb{N}$, we say that a k-CNF^A formula is a k-CNF in n variables $x_1, \dots, x_n$ whose *atoms* are either literals, or they are of the form $A(x_{i_1}, \dots, A_{i_{k'}})$ where $k' \in [k]$ and each $i_j \in [n]$. For example,

$$(x_1 \vee A(x_2, x_3) \vee A(x_3, x_3, x_5)) \wedge (\neg x_2 \vee \neg x_3 \vee A(x_7, x_6, x_7))$$

is a 3-CNF^A formula. Recall that 3-CNF-SAT^A (where we are given a CNF^A formula F^A and are asked if F^A is satisfiable) is NP^A-complete, and its corresponding counting version #3-CNF-SAT^A is #P^A-complete. This definition appeared in [Sch81, GJ93].

Reminder of Proposition 1.8. *For every oracle A, #CNF-SAT^A on formulas with n variables and size* $\text{poly}(n)$ *can be computed in Merlin-Arthur time* $2^{n/2} \cdot \text{poly}(n)$ *with oracle access to the multilinear extension of A over any field of characteristic greater than 2^n (and order at most $2^{\text{poly}(n)}$).*

Proof Sketch. The proposition follows almost directly from the same sort of argument used by Aaronson and Wigderson [AW09] to show that PSPACE = IP algebrizes, applying it to Williams' protocol. Given a #CNF-SAT^A instance F^A on n variables with $\text{poly}(n)$ size, we can think of F^A as an AND of $\text{poly}(n)$ ORs of $\text{poly}(n)$ literals plus copies of the oracle A which take variables as input. We convert F^A into an arithmetic circuit over $\mathbb{F}_p$ where $p > 2^n$ is a prime in the natural way, where the ANDs and ORs are replaced by corresponding multilinear polynomials of degree at most $\text{poly}(n)$, and the copies of oracle A are replaced by calls to the multilinear extension $\tilde{A}$ of A. This results in an arithmetic circuit C of at most $\text{poly}(n)$ degree that agrees with F^A on all Boolean assignments, with the property that C can be evaluated on any

[9] Technically, Merlin speaks before Arthur in a Merlin-Arthur protocol, but note that Merlin could have sent all of his proofs from step 3 prior to steps 1 and 2.

particular assignment in $(\mathbb{F}_p)^n$ in poly(n) time, provided $p < 2^{\text{poly}(n)}$. Note that we are using the fact that we have oracle access to $\tilde{A}$: without it, we would not necessarily be able to evaluate C in poly(n) time.

The Merlin-Arthur protocol then divides the set of variables into two halves, and creates a new arithmetic circuit C' on $n/2$ variables, which equals the sum of $C(x_1, \ldots, x_{n/2}, \vec{a})$ where $\vec{a}$ ranges over all $2^{n/2}$ Boolean assignments to the second half of variables. Merlin tells Arthur a list of values $v_1, \ldots, v_{2^{n/2}} \in \mathbb{F}_p$, and wishes to prove to Arthur that $C'(b_i) = v_i$ for all $i = 1, \ldots, 2^{n/2}$, where $b_1, \ldots, b_{2^{n/2}} \in \{0,1\}^{n/2}$ is a list of all Boolean assignments to the first half of variables (if Merlin can do so, $\sum_i v_i$ will equal the number of satisfying assignments to F^A). This is achieved by first defining "interpolating polynomials" $Q_1, \ldots, Q_{n/2}$ such that for a fixed list of $2^{n/2}$ distinct elements $\alpha_1, \ldots, \alpha_{2^{n/2}} \in \mathbb{F}_p$, we have that $Q_i(\alpha_j)$ outputs the i-th bit of the assignment b_j. Note that each Q_i has degree at most $2^{n/2}$. Merlin sends to Arthur a univariate polynomial $P(y)$ of degree $2^{n/2} \cdot \text{poly}(n)$ representing the circuit C' composed with these Q_i's. Arthur checks P by:

(a) Picking a random point $a \in \mathbb{F}_p$, and confirming that $C'(Q_1(a), \ldots, Q_{n/2}(a)) = P(a)$, in $2^{n/2} \cdot \text{poly}(n)$ time (using the properties of our C and C'), and

(b) Checking for all $i = 1, \ldots, 2^{n/2}$ that $P(\alpha_i) = v_i$ in $2^{n/2} \cdot \text{poly}(n)$ time, using fast univariate polynomial evaluation.

Finally, Arthur concludes that $\sum_{i=1}^{2^{n/2}} v_i$ equals the number of satisfying assignments to F^A. □

Definition 6.1 (DISJ)**.** *In the set-disjointness problem (DISJ), Alice and Bob get n-bit strings x and y, respectively, and their goal is to determine whether $\sum_{i \in [n/2]} x_i \cdot y_i = 0$ holds.*

Recall that in a Merlin-Arthur communication protocol between Alice and Bob: Merlin sends a proof to both Alice and Bob, and then Alice and Bob run a randomized communication protocol given their inputs and the proof from Merlin to decide whether they accept or not. The complexity of this protocol is bounded by the proof length of Merlin plus the maximum number of bits communicated between Alice and Bob.[10]

Now we show that this protocol is actually *optimal* among those which algebrize.

Reminder of Proposition 1.9. *There is an oracle A such that there is* **no** *Merlin-Arthur protocol running in $2^{n/2}/n^{\omega(1)}$ time for 1-UNSAT^A, even for protocols with oracle access to the multilinear extension of A (over any field of order $2^{\text{poly}(n)}$).*

Proof. Our proof directly follows the connection between communication complexity lower bounds and non-algebrizing results that was already described in [AW09]. For completeness, we give a self-contained proof.

Suppose for contradiction that for all oracles A, there is a Merlin-Arthur protocol for instances of 1-UNSAT^A instance on n variables and poly(n) size that runs in $2^{n/2}/n^{\omega(1)}$ time where the protocol has oracle access to $\tilde{A}$, the (unique) multilinear extension of A over $\mathbb{F}_q$ for *some* prime power $q \leq 2^{\text{poly}(n)}$. We will show that DISJ has a Merlin-Arthur *communication* protocol with $o(\sqrt{n})$ communication on n-bit strings, contradicting the known $\sqrt{n}$ lower bound for Merlin-Arthur protocols computing DISJ [Kla03].

Let Alice hold input x and Bob hold input y, each of length $n/2$ (without loss of generality, we assume that n is a power of 2). We think of Alice as holding half the bits of an oracle $A\colon [n/2] \times \{0,1\} \to \{0,1\}$ and Bob holding the other half. More precisely,

$$x = A(1,0)A(2,0)\cdots A(n/2,0) \quad \text{and} \quad y = A(1,1)A(2,1)\cdots A(n/2,1).$$

We want to compute

$$\mathsf{DISJ}(x,y) = \left[\sum_{i \in [n/2]} A(i,0) \cdot A(i,1) = 0\right],$$

where $[P]$ takes value 1 if the statement P is true, and 0 otherwise.

Letting $t = \log_2(n/2)$ and letting our formula be

$$F^A(z_1, \ldots, z_t) = A(z_1, \ldots, z_t, 0) \wedge A(z_1, \ldots, z_t, 1),$$

[10] As in standard Merlin-Arthur protocols, there exists a proof from Merlin making them accept with probability 1 given a yes instance, and they reject every possible proof with high probability given a no instance.

it is clear that $\mathsf{DISJ}(x,y) = \text{1-}\mathsf{UNSAT}^A(F^A)$ (note that F^A is a $\text{1-}\mathsf{SAT}^A$ formula).

By assumption, there is a Merlin-Arthur protocol (with access to the unique multilinear extension $\tilde{A}$) running in time $2^{t/2}/t^{\omega(1)}$ for computing $\text{1-}\mathsf{UNSAT}^A(F^A)$. Let $n_1 = 2^{t/2}/t^{\omega(1)} = \sqrt{n}/(\log n)^{\omega(1)}$. By definition, the algorithm proceeds by guessing n_1 bits, randomly choosing n_1 bits, and then running an n_1-time algorithm that makes at most n_1 calls to $\tilde{A}$.

Alice and Bob compute DISJ as follows. First, they both know the formula F^A (but not necessarily the oracle A). So they just start simulating the MA protocol for $\text{1-}\mathsf{UNSAT}^A(F^A)$ separately. They can obviously simulate the Merlin and Arthur steps in an MA communication protocol, by having "public" nondeterminism of n_1 bits followed by "public" randomness. To simulate the deterministic algorithm making oracle calls, Alice and Bob have to communicate as follows. To handle all n_1 oracle calls, they need to make up to n_1 evaluations of

$$\tilde{A}(a_1, \ldots, a_t, a_{t+1})$$

on given tuples of points a in $\mathbb{F}_q^{t+1}$ (the tuple is determined by all the information computed so far, which both Alice and Bob know). Note that because $\tilde{A}$ is multilinear, we can always write

$$\tilde{A}(a_1, \ldots, a_t, a_{t+1}) = a_{t+1} \cdot A_1(a_1, \ldots, a_t) + (1 - a_{t+1})A_0(a_1, \ldots, a_t)$$

for some multilinear A_0 and A_1.

Now, what are these A_0 and A_1? Well, when we plug in $a_{t+1} = 0$, $\tilde{A} = A_0$, and note the remaining function has a truth table equal to x. Similarly when we plug in $a_{t+1} = 1$, the remaining function has a truth table equal to y, which is just A_1. Therefore Alice can actually compute $A_0(a_1, \ldots, a_t)$ by herself, and Bob can compute $A_1(a_1, \ldots, a_t)$. So to evaluate $\tilde{A}(a_1, \ldots, a_t, a_{t+1})$, the two only have to exchange $O(\log(q))$ bits (the values of A_0 and A_1 on these tuples). Thus they can simulate each query to $\tilde{A}$ using $O(\log(q)) \le \text{poly}(t) \le \text{polylog}(n)$ bits of communication. It follows that they can jointly compute DISJ with only $n_1 \cdot \text{polylog}(n) = o(\sqrt{n})$ communication, contradicting the known $\sqrt{n}$ lower bound for Merlin-Arthur protocols computing DISJ. □

7 Quantified Boolean Formulas

We consider Quantified Boolean Formulas (QBFs) in prenex normal form

$$(Q_1 x_1) \cdots (Q_n x_n)\ F(x_1, \ldots, x_n),$$

where F is an arbitrary propositional formula of size m, preceded by quantifiers of the form $Q_i \in \{\exists, \forall\}$.

Williams [Wil16] gave a 3-round interactive protocol (i.e., an AMA protocol) for QBFs that ran in $O^*(2^{2n/3})$ time. It was asked as an open question [Wil16] whether there is a 2-round Merlin-Arthur protocol for QBFs with $O^*(2^{(1-\varepsilon)n})$ running time for some constant $\varepsilon > 0$. Here we resolve this open problem:

Reminder of Theorem 1.10. *True Quantified Boolean Formulas (TQBF) with n variables and size $m \le 2^n$ can be certified by a Merlin-Arthur protocol running in $2^{4n/5} \cdot \text{poly}(n,m)$ time.*

Our new protocol follows the basic outline of Williams' earlier AMA protocol [Wil16, Section 4], with several key differences we highlight in the proof.

We will prove the following lemma.

Lemma 7.1. *Let*

$$\phi = (Q_1 x_1) \cdots (Q_n x_n)\ F(x_1, \ldots, x_n)$$

be a QBF. Suppose there exist integers $1 \le k \le \ell \le n$ such that the last ℓ quantifiers, $Q_{n-\ell+1}, \ldots, Q_n$, contain at most k universal quantifiers. Then

- *if ϕ is a true QBF, we can certify ϕ in Merlin-Arthur time $(2^{n+k-\ell} + 2^{\ell}) \cdot \text{poly}(n,m)$, and*
- *if ϕ is a false QBF, we can refute ϕ in Merlin-Arthur time $(2^{n+2k-\ell} + 2^{\ell+k}) \cdot \text{poly}(n,m)$.*

Before proving Lemma 7.1, we show that it implies the claimed QBF protocol.

Proof of Theorem 1.10 using Lemma 7.1. Let $\phi = (Q_1 x_1) \cdots (Q_n x_n)\ F(x_1, \ldots, x_n)$ be a true QBF to certify. Let $0 < \alpha < \delta < 1$ be two constant parameters to be determined later. As in [Wil16], we divide into two cases depending on the number of universal quantifiers contained in the last δn quantifiers, $(Q_{n-\delta n+1} x_{n-\delta n+1}) \cdots (Q_n x_n)$.

- Case 1: The last δn quantifiers contain at most αn universal quantifiers.

  In this case, the first item in Lemma 7.1 implies a Merlin-Arthur protocol in $(2^{(1+\alpha-\delta)n} + 2^{\delta n}) \cdot \mathrm{poly}(n, m)$ time.

- Case 2: The last δn quantifiers contain more than αn universal quantifiers.

  In this case, we prove that $\neg\phi$ is false using the second item in Lemma 7.1. Since the last δn quantifiers in $\neg\phi$ has more than αn existential quantifiers and less than $\delta n - \alpha n$ universal quantifiers, applying Lemma 7.1 gives a Merlin-Arthur protocol in $(2^{(1+\delta-2\alpha)n} + 2^{(2\delta-\alpha)n}) \cdot \mathrm{poly}(n, m)$ time.

Setting $\alpha = 2/5, \delta = 3/5$ yields a $2^{4n/5} \cdot \mathrm{poly}(n, m)$ time Merlin-Arthur protocol as claimed. □

To complete the argument, it remains to prove Lemma 7.1.

Proof of Lemma 7.1. First, we apply the same strategy as in [Wil16]. Convert the propositional formula F to an equivalent arithmetic formula P of $\mathrm{poly}(m)$ degree and size, by replacing $A \wedge B$ with $A \cdot B$ and replacing $A \vee B$ with $A + B - A \cdot B$. Note that P outputs 0 or 1 on every Boolean input. Then, we convert the subformula

$$\phi'(x_1, \ldots, x_{n-\ell}) = (Q_{n-\ell+1} x_{n-\ell+1}) \cdots (Q_n x_n)\ P(x_1, \ldots, x_n)$$

into an arithmetic formula P', by replacing each $(\exists x_i)$ with a sum over $x_i \in \{0, 1\}$, and each $(\forall x_i)$ with a product over $x_i \in \{0, 1\}$. Note that for every $a_1, \ldots, a_{n-\ell} \in \{0, 1\}^{n-\ell}$, $P'(a_1, \ldots, a_{n-\ell})$ evaluates to a positive integer if the subformula $\phi'(a_1, \ldots, a_{n-\ell})$ is true, and evaluates to zero if $\phi'(a_1, \ldots, a_{n-\ell})$ is false.

Note that P' has a depth-ℓ binary tree structure with each leaf being a copy of P. The size of P' is at most $2^\ell \cdot \mathrm{poly}(m)$, and the degree of P' is at most $2^k \cdot \mathrm{poly}(m)$, since there are at most k layers of multiplication gates in this binary tree. For every Boolean input $a_1, \ldots, a_{n-\ell} \in \{0, 1\}^{n-\ell}$, observe that the output of $P'(a_1, \ldots, a_{n-\ell})$ is a non-negative integer no larger than $(2^n \cdot m)^{O(2^k)}$.

Now we separately consider the two scenarios.

Case 1: To prove ϕ is true. In this case, Merlin sends Arthur a prime p from the interval $[2, 2^{2n^2} \cdot m]$, such that, for every $a_1, \ldots, a_{n-\ell} \in \{0, 1\}$ with $P'(a_1, \ldots a_{n-\ell})$ being a positive integer (over $\mathbb{Z}$), $P'(a_1, \ldots a_{n-\ell}) \bmod p$ is also non-zero (over $\mathbb{F}_p$). The existence of such prime p was already proved in [Wil16, Section 4] using a standard argument by considering the number of prime factors of $P'(a_1, \ldots a_{n-\ell})$ and applying a union bound.

Then, Merlin and Arthur perform Williams' [Wil16] batch-evaluation protocol (Theorem 2.2) over the field $\mathbb{F}_p$: Merlin sends the correct values of $P'(a_1, \ldots a_{n-\ell}) \bmod p$ over all $a_1, \ldots, a_{n-\ell} \in \{0, 1\}$, together with a proof of length $\tilde{O}(2^{n-\ell} \cdot \deg(P') \cdot \log(p)) \le 2^{n-\ell+k} \cdot \mathrm{poly}(n, m)$. Then Arthur verifies the proof

$$\tilde{O}(2^{n-\ell} \cdot \deg(P') + \mathrm{size}(P')) \cdot \mathrm{poly}\log(p) \le (2^{n-\ell+k} + 2^\ell) \cdot \mathrm{poly}(n, m)$$

time. Finally, Arthur uses these values to certify

$$\phi = (Q_1 x_1) \cdots (Q_{n-\ell} x_{n-\ell})\ \phi'(x_1, \ldots, x_{n-\ell})$$

is true in $O(2^{n-\ell})$ time.

The only concern is that the prime p sent by Merlin might not satisfy the required condition: there could exist some $(a_1, \ldots, a_{n-\ell}) \in \{0, 1\}^{n-\ell}$ where $P'(a_1, \ldots, a_{n-\ell})$ is non-zero over $\mathbb{Z}$ but is zero over $\mathbb{F}_p$, so that Arthur will be evaluating $\phi = (Q_1 x_1) \cdots (Q_{n-\ell} x_{n-\ell}) \phi'(x_1, \ldots, x_{n-\ell})$ based on incorrect values of $\phi'(a_1, \ldots, a_{n-\ell})$. However, Merlin is not able to cheat by doing this, since the value of ϕ is monotone increasing in the values of $\phi'(a_1, \ldots, a_{n-\ell})$, and modifying some of these values from true to false will never change the value of ϕ from false to true.

We remark that the only difference of this protocol from the previous AMA protocol [Wil16] is that we let Merlin send the prime p, whereas [Wil16] let Arthur send a random p (which satisfies the required condition with high probability), costing an extra round of interaction.

Case 2: To prove ϕ is false. Note that the previous protocol for the "ϕ is true" case no longer applies here, since Merlin would be able to cheat by picking a prime p that makes many of the positive integers $P'(a_1, \ldots, a_{n-\ell})$ vanish in $\mathbb{F}_p$.

Recall that these positive integers $P'(a_1, \ldots, a_{n-\ell})$ are upper bounded by $(2^n \cdot m)^{O(2^k)}$. Instead of picking a single prime p for the protocol, Merlin picks s distinct primes $p_1 < p_2 < \cdots < p_s$ so that their product $p_1 p_2 \cdots p_s$ is larger than this upper bound. In this way, by Chinese Remainder Theorem we can ensure that, every positive integer $P'(a_1, \ldots, a_{n-\ell})$ is non-zero mod p_j for at least one $1 \le j \le s$. Then, we can simply run the previous protocol for *every* p_j $(1 \le j \le s)$, in total time $s \cdot (2^{n-\ell+k} + 2^\ell) \cdot \mathrm{poly}(n, m, \log p_s)$.

By choosing the smallest s primes $p_1 < \cdots < p_s$ such that $p_1 p_2 \ldots p_s > (2^n \cdot m)^{\Omega(2^k)}$, we can ensure the above algorithm works with parameter choices $s \le 2^k \cdot \mathrm{poly}(n \log m)$, and $p_s \le O(s \cdot \log s)$ by the prime number theorem. Hence, the total time complexity is $(2^{n-\ell+2k} + 2^{k+\ell}) \cdot \mathrm{poly}(n, m)$. □

8 Open Questions

There remain many interesting open problems concerning the nondeterministic and Merlin-Arthur complexity of problems in fine-grained complexity. A few questions which are particularly relevant to our work are highlighted below.

- Is there a faster Merlin-Arthur protocol for solving Subset Sum? Corollary 1.2 gives a $2^{n/3} \cdot \mathrm{poly}(n)$ time protocol using a linear-time protocol for 3-SUM. There are deterministic algorithms which solve Subset Sum in $2^{n/2} \cdot \mathrm{poly}(n)$ time, so it seems plausible (by analogy with Williams' Merlin-Arthur protocol for counting satisfying assignments) that one could achieve a quadratic improvement over this speed and certify that an instance of Subset Sum has no solution in $2^{n/4} \cdot \mathrm{poly}(n)$ Merlin-Arthur time.

  Is there a $2^{n/4} \cdot \mathrm{poly}(n)$-time Merlin-Arthur protocol for certifying that a Subset Sum instance has no solutions? Such a protocol would exist, for example, if there was a linear time Merlin-Arthur protocol for 4-SUM.

- Our Merlin-Arthur protocol for APSP presented in Corollary 1.6 yields near-optimal protocols for many related graph problems on dense graphs. Can similar improvement be achieved on sparse graphs? For example, is there a constant $\epsilon > 0$ such that there exists an $n^{2-\varepsilon}$-time Merlin-Arthur protocol for computing the diameter of a graph with n nodes and $O(n)$ edges?

- Our Merlin-Arthur protocol from Theorem 1.5 counts the number of zero-weight 5-cliques in $\tilde{O}(nm)$ time, which could be as large as n^3 for dense graphs. Is there an $\tilde{O}(n^{2.5})$-time Merlin-Arthur protocol for this problem (or, the problem of certifying that no such clique exists)? Achieving the constant 2.5 in the exponent of the runtime would yield a quadratic improvement over the deterministic n^5-time algorithm for zero-weight 5-clique.

- Is there a $2^{n/2} \cdot \mathrm{poly}(n)$-time Merlin-Arthur protocol for certifying True Quantified Boolean Formulas? Currently, the best Merlin-Arthur protocol runs in $2^{4n/5} \cdot \mathrm{poly}(n)$ time, and the best AMA protocol takes $2^{2n/3} \cdot \mathrm{poly}(n)$ time [Wil16].

- Suppose the Nonuniform NSETH holds (e.g., there is no infinite family of nondeterministic circuits $\{C_n\}$ of size 1.999^n that correctly solves CNF unsatisfiability on n-variable formulas of $\mathrm{poly}(n)$ size). This would also imply a lower bound for (one-round) Arthur-Merlin protocols. Would this hypothesis further imply interesting Arthur-Merlin communication lower bounds? Proving non-trivial lower bounds on two-party Arthur-Merlin communication complexity is an infamously difficult problem; as far as we know, two-party Arthur-Merlin communication could be very powerful (see for example [Kla11, GPW18]). Perhaps an interesting lower bound for the Disjointness problem follows from Nonuniform NSETH?

- Given a universe $U = \{1, \ldots, n\}$ of n elements, a family $\mathcal{F}$ of subsets of U, and a target integer t, the #Set Cover problem is the task of computing how many choices of t sets from $\mathcal{F}$ have the property that their union equals U. Similarly, the #Exact Cover is the task of counting how choices of t *pairwise disjoint* sets from $\mathcal{F}$ have their union equal to U.

Both these problems can be solved deterministically in $2^n \cdot \text{poly}(n)$ time. However, in the Merlin-Arthur setting, although there is a $\left(2^{n/2} + |\mathcal{F}|\right) \text{poly}(n)$-time protocol for solving #Exact Cover, the fastest known protocol for solving #Set Cover takes $2^{n/2}|\mathcal{F}| \cdot \text{poly}(n)$ time [BK16]. Is there a faster Merlin-Arthur protocol for #Set Cover, or is #Set Cover truly harder than #Exact Cover in the Merlin-Arthur setting for families consisting of $2^{\Omega(n)}$ sets?

- To what extent can our Merlin-Arthur protocols be derandomized to obtain better nondeterministc algorithms for fine-grained problems? For example, derandomizing our protocol for 3-SUM without a loss in the running time would imply a nondeterministic derandomization of Freivald's verification algorithm for Boolean Matrix Multiplication [Kün18, Theorem 1.1] and answer an open question raised by [Kün18]. Finding faster nondeterministic verifiers in this way may also lead to new barriers in deterministic fine-grained reductions between problems [CGI+16].

For all of the problems discussed above, evidence against the existence of a better algorithm or protocol (via conditional hardness results) would also be interesting.

Acknowledgements

We thank Virginia Vassilevska Williams for offering helpful comments on early versions of our arguments. We thank Jesper Nederlof for answering a question about his Merlin-Arthur protocol for Subset Sum [Ned21].

References

[ABV18] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. If the current clique algorithms are optimal, so is Valiant's parser. *SIAM J. Comput.*, 47(6):2527–2555, 2018. 3

[AGI+19] Amir Abboud, Loukas Georgiadis, Giuseppe F. Italiano, Robert Krauthgamer, Nikos Parotsidis, Ohad Trabelsi, Przemysław Uznański, and Daniel Wolleb-Graf. Faster Algorithms for All-Pairs Bounded Min-Cuts. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, pages 7:1–7:15, 2019. 3

[AGV15] Amir Abboud, Fabrizio Grandoni, and Virginia Vassilevska Williams. Subcubic equivalences between graph centrality problems, APSP and diameter. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2015)*, pages 1681–1697, 2015. 1

[AKKN16] Per Austrin, Petteri Kaski, Mikko Koivisto, and Jesper Nederlof. Dense Subset Sum may be the hardest. In *Proceedings of the 33rd Symposium on Theoretical Aspects of Computer Science (STACS 2016)*, pages 13:1–13:14, 2016. 2

[AV21] Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA 2021)*, pages 522–539, 2021. 3

[AW09] Scott Aaronson and Avi Wigderson. Algebrization: A new barrier in complexity theory. *ACM Trans. Comput. Theory*, 1(1):2:1–2:54, 2009. 3, 4, 15, 16

[AWY15] Amir Abboud, Richard Ryan Williams, and Huacheng Yu. More applications of the polynomial method to algorithm design. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2015)*, pages 218–230, 2015. 1

[BBB19] Enric Boix-Adserà, Matthew S. Brennan, and Guy Bresler. The average-case complexity of counting cliques in Erdős-Rényi hypergraphs. In *Proceedings of the 60th IEEE Annual Symposium on Foundations of Computer Science (FOCS 2019)*, pages 1256–1280, 2019. 1

[BCD$^+$14] David Bremner, Timothy M. Chan, Erik D. Demaine, Jeff Erickson, Ferran Hurtado, John Iacono, Stefan Langerman, Mihai Pătraşcu, and Perouz Taslakian. Necklaces, convolutions, and X+Y. *Algorithmica*, 69(2):294–314, 2014. 2

[BDT16] Arturs Backurs, Nishanth Dikkala, and Christos Tzamos. Tight hardness results for maximum weight rectangles. In *Proceedings of the 43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, pages 81:1–81:13, 2016. 1

[BK16] Andreas Björklund and Petteri Kaski. How proofs are prepared at Camelot: Extended abstract. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing (PODC 2016)*, pages 391–400, 2016. 1, 3, 20

[BPVZ14] Andreas Björklund, Rasmus Pagh, Virginia Vassilevska Williams, and Uri Zwick. Listing triangles. In *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming (ICALP 2014), Part I*, pages 223–234, 2014. 14

[BRSV17] Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. Average-case fine-grained hardness. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2017)*, pages 483–496, 2017. 1

[BRSV18] Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. Proofs of work from worst-case assumptions. In *Proceedings of the 38th Annual International Cryptology Conference (CRYPTO 2018), Part I*, pages 789–819, 2018. 1

[CG19] Amit Chakrabarti and Prantar Ghosh. Streaming verification of graph computations via graph structure. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2019)*, volume 145 of *LIPIcs*, pages 70:1–70:20, 2019. 1

[CGI$^+$16] Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science (ITCS 2016)*, pages 261–270, 2016. 1, 2, 6, 8, 20

[CGT20] Amit Chakrabarti, Prantar Ghosh, and Justin Thaler. Streaming verification for graph problems: Optimal tradeoffs and nonlinear sketches. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020)*, volume 176 of *LIPIcs*, pages 22:1–22:23, 2020. 1

[CMWW19] Marek Cygan, Marcin Mucha, Karol Węgrzycki, and Michał Włodarczyk. On problems equivalent to (min, +)-convolution. *ACM Trans. Algorithms*, 15(1):14:1–14:25, 2019. 2, 11

[CW19] Lijie Chen and Ryan Williams. An equivalence class for orthogonal vectors. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2019)*, pages 21–40. SIAM, 2019. 1

[CW21] Timothy M. Chan and R. Ryan Williams. Deterministic APSP, Orthogonal Vectors, and more: Quickly derandomizing Razborov-Smolensky. *ACM Trans. Algorithms*, 17(1):2:1–2:14, 2021. 1, 2

[DLV20] Mina Dalirrooyfard, Andrea Lincoln, and Virginia Vassilevska Williams. New techniques for proving fine-grained average-case hardness. In *Proceedings of the 61st IEEE Annual Symposium on Foundations of Computer Science (FOCS 2020)*, pages 774–785, 2020. 1

[Fid72] Charles M. Fiduccia. Polynomial evaluation via the division algorithm: The fast Fourier transform revisited. In *Proceedings of the 4th Annual ACM Symposium on Theory of Computing (STOC 1972)*, pages 88–93, 1972. 5

[GIKW19] Jiawei Gao, Russell Impagliazzo, Antonina Kolokolova, and Ryan Williams. Completeness for first-order properties on sparse structures with algorithmic applications. *ACM Trans. Algorithms*, 15(2):23:1–23:35, 2019. 1

[GJ93] Judy Goldsmith and Deborah Joseph. Relativized isomorphisms of NP-complete sets. *Comput. Complex.*, 3:186–205, 1993. 15

[GO95] Anka Gajentaan and Mark H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Comput. Geom.*, 5:165–185, 1995. 10

[GPW18] Mika Göös, Toniann Pitassi, and Thomas Watson. The landscape of communication complexity classes. *Comput. Complex.*, 27(2):245–304, 2018. 19

[GR18a] Oded Goldreich and Guy N. Rothblum. Counting t-cliques: Worst-case to average-case reductions and direct interactive proof systems. In *Proceedings of the 59th IEEE Annual Symposium on Foundations of Computer Science (FOCS 2018)*, pages 77–88, 2018. 1

[GR18b] Oded Goldreich and Guy N. Rothblum. Simple doubly-efficient interactive proof systems for locally-characterizable sets. In *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, volume 94 of *LIPIcs*, pages 18:1–18:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. 1

[GR20a] Oded Goldreich and Guy N. Rothblum. Constant-round interactive proof systems for AC0[2] and NC1. In *Computational Complexity and Property Testing - On the Interplay Between Randomness and Computation*, volume 12050 of *Lecture Notes in Computer Science*, pages 326–351. Springer, 2020. 1

[GR20b] 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. 1

[Hor72] Ellis Horowitz. A fast method for interpolation using preconditioning. *Inf. Process. Lett.*, 1(4):157–163, 1972. 5

[HS74] Ellis Horowitz and Sartaj Sahni. Computing partitions with applications to the knapsack problem. *J. ACM*, 21(2):277–292, 1974. 2

[HS21] Shuichi Hirahara and Nobutaka Shimizu. Nearly optimal average-case complexity of counting bicliques under SETH. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA 2021)*, pages 2346–2365, 2021. 1

[IKK09] Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. An axiomatic approach to algebrization. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC 2009)*, pages 695–704, 2009. 3

[IP01] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. 4, 14

[IR78] Alon Itai and Michael Rodeh. Finding a minimum circuit in a graph. *SIAM J. Comput.*, 7(4):413–423, 1978. 3

[JV16] Zahra Jafargholi and Emanuele Viola. 3SUM, 3XOR, triangles. *Algorithmica*, 74(1):326–343, 2016. 11

[Kla03] Hartmut Klauck. Rectangle size bounds and threshold covers in communication complexity. In *Proceedings of the 18th Annual IEEE Conference on Computational Complexity (CCC 2003)*, pages 118–134, 2003. 16

[Kla11] Hartmut Klauck. On Arthur Merlin games in communication complexity. In *Proceedings of the 26th Annual IEEE Conference on Computational Complexity (CCC 2011)*, pages 189–199, 2011. 19

[KU11] Kiran S. Kedlaya and Christopher Umans. Fast polynomial factorization and modular composition. *SIAM J. Comput.*, 40(6):1767–1802, 2011. 6

[Kün18] Marvin Künnemann. On Nondeterministic Derandomization of Freivalds' Algorithm: Consequences, Avenues and Algorithmic Progress. In *Proceedings of the 26th Annual European Symposium on Algorithms (ESA 2018)*, pages 56:1–56:16, 2018. 20

[LG14] François Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation (ISSAC 2014)*, pages 296–303, 2014. 3

[LLV19] Rio LaVigne, Andrea Lincoln, and Virginia Vassilevska Williams. Public-key cryptography in the fine-grained setting. In *Proceedings of the 39th Annual International Cryptology Conference (CRYPTO 2019), Part III*, pages 605–635, 2019. 1

[LVW18] Andrea Lincoln, Virginia Vassilevska Williams, and R. Ryan Williams. Tight hardness for shortest cycles and paths in sparse graphs. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2018)*, pages 1236–1252, 2018. 1, 3

[Ned17] Jesper Nederlof. A short note on Merlin-Arthur protocols for subset sum. *Inf. Process. Lett.*, 118:15–16, 2017. 2, 7

[Ned21] Jesper Nederlof. Personal communication, 2021. 2, 20

[NP85] Jaroslav Nešetřil and Svatopluk Poljak. On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae*, 26(2):415–419, 1985. 3

[Păt10] Mihai Pătraşcu. Towards polynomial lower bounds for dynamic problems. In *Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC 2010)*, pages 603–610, 2010. 1, 2, 14

[RRR18] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Efficient batch verification for UP. In *Proceedings of the 33rd Computational Complexity Conference (CCC 2018)*, pages 22:1–22:23, 2018. 1

[RRR21] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. *SIAM J. Comput.*, 50(3), 2021. 1

[Sch81] Uwe Schöning. A note on complete sets for the polynomial-time hierarchy. *SIGACT News*, 13(1):30–34, January 1981. 15

[Vas12] Virginia Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. In *Proceedings of the 44th Symposium on Theory of Computing Conference (STOC 2012)*, pages 887–898, 2012. 3

[Vas18] Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the International Congress of Mathematicians*, pages 3447–3487. World Scientific, 2018. 1

[VW13] Virginia Vassilevska Williams and Ryan Williams. Finding, minimizing, and counting weighted subgraphs. *SIAM J. Comput.*, 42(3):831–854, 2013. 2, 3, 11

[VW18] Virginia Vassilevska Williams and R. Ryan Williams. Subcubic equivalences between path, matrix, and triangle problems. *J. ACM*, 65(5), August 2018. 1, 2, 3, 14

[VWWY15] Virginia Vassilevska Williams, Joshua R. Wang, Richard Ryan Williams, and Huacheng Yu. Finding four-node subgraphs in triangle time. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2015)*, pages 1671–1680, 2015. 13

[VX20] Virginia Vassilevska Williams and Yinzhan Xu. Monochromatic triangles, triangle listing and APSP. In *Proceedings of the 61st IEEE Annual Symposium on Foundations of Computer Science (FOCS 2020)*, pages 786–797, 2020. 14

[Wil16] Richard Ryan Williams. Strong ETH breaks with Merlin and Arthur: Short non-interactive proofs of batch evaluation. In *Proceedings of the 31st Conference on Computational Complexity (CCC 2016)*, pages 2:1–2:17, 2016. 1, 2, 3, 4, 5, 6, 14, 17, 18, 19

[Wil18] R. Ryan Williams. Faster all-pairs shortest paths via circuit complexity. *SIAM J. Comput.*, 47(5):1965–1985, 2018. 2

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il