The Complexity of Linear Dependence Problems in Vector Spaces

Arnab Bhattacharyya[1] Piotr Indyk[2] David P. Woodruff[3] Ning Xie[4]

[1]MIT CSAIL.* Work done in part at IBM Almaden.

[2]MIT CSAIL.

[3]IBM Almaden.

[4] MIT CSAIL.

abhatt@mit.edu indyk@mit.edu dpwoodru@us.ibm.com ningxie@csail.mit.edu

Abstract: We study extensions of the k-SUM problem to vector spaces over finite fields. Given a subset $S \subseteq \mathbb{F}_q^n$ of size $r \leq q^n$, an integer $k, 2 \leq k \leq n$, and a vector $v \in (\mathbb{F}_q^k \setminus \{0\})^k$, we define the TARGETSUM problem to be the problem of finding k elements $x_{i_1}, \ldots, x_{i_k} \in S$ for which $\sum_{j=1}^k v_j x_{i_j} = z$, where z may either be an input or a fixed vector. We also study a variant of this, where instead of finding $x_{i_1}, \ldots, x_{i_k} \in S$ for which $\sum_{j=1}^k v_j x_{i_j} = z$, we require that z be in $\mathrm{span}(x_{i_1}, \ldots, x_{i_k})$, which we call the (k,r)-LINDEPENDENCE$_q$ problem.

These problems are natural generalizations of well-studied problems that occur in coding theory and property testing. Indeed, the (k,r)-LINDEPENDENCE$_q$ problem is just the MAXIMUM LIKELIHOOD DECODING problem for linear codes. Also, in the TARGETSUM problem, if instead of general z we require $z = 0^n$, then this is the WEIGHT DISTRIBUTION problem for linear codes. In property testing, these problems have been implicitly studied in the context of testing linear-invariant properties.

We give nearly optimal bounds for TARGETSUM and (k,r)-LINDEPENDENCE$_q$ for every r, k, and constant q. Namely, assuming 3-SAT requires exponential time, we show that any algorithm for these problems must run in $\min(2^{\Theta(n)}, r^{\Theta(k)})$ time. Moreover, we give deterministic upper bounds that match this complexity, up to constant factors in the exponent. Our lower bound strengthens and simplifies an earlier $\min(2^{\Theta(n)}, r^{\Omega(k^{1/4})})$ lower bound for both the MAXIMUM LIKELIHOOD DECODING and WEIGHT DISTRIBUTION problems.

We also give upper and lower bounds for variants of these problems, e.g., for the problem for which we must find $x_{i_1}, \ldots, x_{i_k}$ for which $z \in \mathrm{span}(x_{i_1}, \ldots, x_{i_k})$ but z is not spanned by any proper subset of these vectors, and for the counting version of this problem.

Keywords: Coding theory; Computational Geometry, k-Sum

1 Introduction

We study the computational complexity of algorithms that test if linear combinations of certain-sized subsets of a set of input vectors equal a desired target vector. This is a fundamental problem with applications to coding theory, computational geometry, and property testing.

The special case when the field is $\mathbb{R}$, there is only a single dimension, and one wants to find a sum of k numbers that equals 0 is the well-known k-SUM problem. Many problems, especially in computational geometry, have been shown to be k-SUM hard for certain k; see, for example, the works of [20–22, 25, 38, 40]. Some problems known to be 3-SUM hard include 3-POINTS-ON-LINE, MINIMIUM-AREA-TRIANGLE, SEPARATOR, STRIPS-COVER-BOX, TRIANGLES-COVER-TRIANGLE, PLANAR-MOTION-PLANNING, DIHEDRAL-ROTATION, and POLYGON-CONTAINMENT; see the survey by King [31]. As stated in [5], the body of work on 3-SUM "is perhaps the most successful attempt at understanding the complexity inside P (polynomial time)." We believe the study of the extension of this problem to vector spaces over finite fields will likewise result in a deeper understanding of the complexity of many other problems.

Let $\mathbb{F} = \mathbb{F}_q$ be the finite field of q elements and let n be a natural number. We assume that q is a constant independent of n. The main problems we study are the following.

Let $k, r : \mathbb{Z}^+ \to \mathbb{Z}^+$ be functions.

Definition 1 *The (k,r)-ZEROSUM$_q$ problem takes as input $r(n)$ many elements $x_1, \ldots, x_{r(n)} \in \mathbb{F}_q^n$ and checks if*

*Supported in part by a DOE Computational Science Graduate Fellowship and NSF Awards 0514771, 0728645, and 0732334.

there exist $x_{i_1}, \ldots, x_{i_{k(n)}}$ such that $i_1, \ldots, i_{k(n)} \in [r(n)]$ and $x_{i_1} + \cdots + x_{i_k} = 0$.

The (k, r)-TARGETSUM$_q$ problem takes as input $r(n)$ elements $x_1, \ldots, x_{r(n)} \in \mathbb{F}_q^n$ and $z \in \mathbb{F}_q^n$ and checks if there exist $x_{i_1}, \ldots, x_{i_{k(n)}}$ such that $i_1, \ldots, i_{k(n)} \in [r(n)]$ and $x_{i_1} + \cdots + x_{i_{k(n)}} = z$.

We also study the following slight extension to general linear combinations of input vectors. Let $v = (v_1, \ldots, v_{k(n)}) \in (\mathbb{F}_q \backslash \{0\})^{k(n)}$ be a family of vectors.

Definition 2 *For every $n \geq 1$, the[1] (k, r, v)-ZEROSUM$_q$ problem takes as input $r(n)$ elements $x_1, \ldots, x_{r(n)} \in \mathbb{F}_q^n$ and checks if there exist $x_{i_1}, \ldots, x_{i_k}$ such that $i_1, \ldots, i_k \in [r(n)]$ and $v_1 x_{i_1} + \cdots + v_k x_{i_k} = 0$.*

For every $n \geq 1$, the (k, r, v)-TARGETSUM$_q$ problem takes as input $r(n)$ elements $x_1, \ldots, x_{r(n)} \in \mathbb{F}_q^n$ and $z \in \mathbb{F}_q^n$ and checks if there exist $x_{i_1}, \ldots, x_{i_k}$ such that $i_1, \ldots, i_k \in [r(n)]$ and $v_1 x_{i_1} + \cdots + v_k x_{i_k} = z$.

Notice that for $q = 2$, (k, r, v)-ZEROSUM$_q$ and (k, r, v)-TARGETSUM$_q$ coincide with (k, r)-ZEROSUM$_q$ and (k, r)-TARGETSUM$_q$, respectively. We also consider a related problem when it is more useful to look at the span of vectors than it is to fix a single combination v.

Definition 3 *The (k, r)-LINDEPENDENCE$_q$ problem takes as input r elements $x_1, \ldots, x_{r(n)} \in \mathbb{F}_q^n$ and $z \in \mathbb{F}_q^n$ and checks if there exist $x_{i_1}, \ldots, x_{i_{k(n)}}$ such that $z \in \operatorname{span}(x_{i_1}, \ldots, x_{i_{k(n)}})$.*

Finally, we consider a variant of this problem which requires the linear dependence to be *minimal*. Vectors $x_1, \ldots, x_k$ are said to be minimally linearly dependent if $0 \in \operatorname{span}(x_1, \ldots, x_k)$, but 0 cannot be written as a non-trivial linear combination of any proper subset of $\{x_1, \ldots, x_k\}$.

Definition 4 *The (k, r)-MINLINDEPENDENCE$_q$ problem takes as input $r(n)$ elements $x_1, \ldots, x_{r(n)} \in \mathbb{F}_q^n$ and checks if there exist $x_{i_1}, \ldots, x_{i_{k(n)}}$ that are minimally linearly dependent.*

Notice that in all of the above problems, we do not require the $i_1, \ldots, i_k$ in the solution to be distinct. To understand how these problems are related to existing coding-theoretic problems, consider first the (k, r)-LINDEPENDENCE$_q$ problem. This is just the MAXIMUM LIKELIHOOD DECODING problem for linear codes, that is, the problem of determining the most likely transmitted codeword given a certain received word. This problem is well-studied [6, 18, 19, 23, 28, 42, 46]. To see the connection, if the columns of an $n \times r(n)$ parity check matrix are $x_1, \ldots, x_{r(n)}$, and the received word is z, then if there are $k(n)$ elements $x_{i_1}, \ldots, x_{i_{k(n)}}$ whose span contains z, then there is a codeword that if corrupted in at most $k(n)$ positions, equals z.

Now consider the (k, r)-ZEROSUM$_2$ problem. Let A be the $n \times r$ matrix whose columns correspond to the input vectors to this problem. Consider the code $C = \{x \mid Ax = 0\}$. Then C is of dimension at least $r - n$, and the (k, r)-ZEROSUM$_q$ problem has a solution iff there is a codeword of weight k. This is the WEIGHT DISTRIBUTION problem for linear codes in coding theory, a problem studied in [6, 18].

In the property testing literature, these problems have been studied in the context of testing linear-invariant properties [30], though the model differs from ours in the sense that the input set is promised to either contain a certain linear dependence or to be far from a set that does. In analogy to triangles in graphs, we define the *triangles* in a set $S \subseteq \mathbb{F}_q^n$ as the triples $\langle x_1, x_2, x_3 \rangle$ such that $x_1, x_2, x_3 \in S$ and $x_1 + x_2 + x_3 = 0$. Similarly, for $k \geq 3$ we define the *k-cycles* [7] in a set $S \subseteq \mathbb{F}_q^n$ to be the k-tuples $\langle x_1, x_2, \ldots, x_k \rangle$ such that each $x_i \in S$ for $i = 1, 2, \ldots, k$ and $x_1 + x_2 + \cdots + x_k = 0$. Green [27] showed that for constant k, one can distinguish, in constant time, between the case when the input set is free from k-cycles and the case when a constant fraction of the elements of the set need to be removed in order to make it free. This result has been generalized in several directions [7, 8, 32, 33, 44]. In particular, Shapira [44] and Kráľ', Serra and Vena [32] independently showed that testing whether a set S is free from containing tuples $x = (x_1, \ldots, x_k) \in S^k$ satisfying $Mx = b$ (where M is a constant-sized matrix over $\mathbb{F}_q$ with k columns and b is a vector over $\mathbb{F}_q^n$), or whether S is far from being such a set, can also be done in constant time. Our work can be viewed as addressing the classical versions of these problems.

The problems we study are also similar to those of finding subgraphs inside of graphs. For example, finding solutions to $x_1 + x_2 + \cdots + x_k = 0$ in a set $S \subseteq \mathbb{F}_q^n$ and finding k-cliques in a graph both require finding k items from the input (elements and vertices, respectively) that satisfy a given constraint. There has been a lot of algorithmic work on the problem of finding subgraphs [1–4, 15, 26, 43, 47, 48]. The best known algorithm for finding triangles [4] runs in time $O(\min(|E|^{2\omega/(\omega+1)}, n^\omega))$ where ω is the matrix multiplication exponent, and the best known algorithm for finding k-cliques in a graph

[1] We allow a slight abuse of notation in allowing v to refer both to the family of vectors and each vector itself.

with n vertices, due to Nešetřil and Poljak [39], runs in time $O(n^{.792k})$. It is not known whether algorithms for either of these problems running in time $O(n^2)$ can be ruled out, although an algorithm for k-clique running in time $n^{o(k)}$ would imply a subexponential algorithm for 3-SAT [11, 49]. In contrast, we show that the situation for our linear algebraic problems is much clearer. We can show nearly tight upper and lower bounds based on standard complexity assumptions. Additionally, our upper bounds are relatively stronger, essentially because, while the graph algorithms use fast matrix multiplication which runs in time $n^{2+0.376\ldots}$ for two n-by-n matrices, we can use fast convolution which runs in time $N^{1+o(1)}$ for two real-valued functions over a finite field of order N.

1.1 Results

Assuming 3-SAT cannot be solved in sub-exponential time, we resolve (up to polynomial factors) the time complexity of the problems:

- (k, r)-ZeroSum$_q$,
- (k, r)-TargetSum$_q$,
- (k, r, v)-ZeroSum$_q$,
- (k, r, v)-TargetSum$_q$, and the
- (k, r)-LinDependence$_q$

problems. Namely, we show that any deterministic algorithm must run in $\min(r^{\Theta(k)}, 2^{\Theta(n)})$ time, and we give a deterministic algorithm running in this amount of time to solve these problems. Our lower bound also holds for randomized algorithms, provided we assume that 3-SAT cannot be solved in sub-exponential time by a randomized algorithm. The complexity assumption we use is the well-known Exponential Time Hypothesis conjectured by Impagliazzo, Paturi, and Zane [29], and used in a number of papers to establish hardness results [12, 13, 24, 34–36]. It is known that this assumption is equivalent to the assumption that d-SAT cannot be solved in sub-exponential time for some constant $d \geq 3$.

Our lower bound strengthens and simplifies the previous lower bound for both the Maximum Likelihood Decoding and Weight Distribution problems [18]. In that paper, the authors start with the Independent Set problem of size k on graphs of n vertices, and produce an instance of what is called the Perfect Code problem [16, 17] with parameter k^2 on graphs containing n^2 vertices. Then, the authors obtain a more "robust" version of Perfect Code, with certain properties of every dominating set in the instance. The new instance has parameter k^4 and the corresponding graphs contain at least n^2 vertices, and is used to derive hardness results for coding-theoretic problems. This last step is Theorem 1 in [18] and is complicated, the proof introducing a number of gadgets and spanning about eight pages. Due to the chain of reductions, this implies that the lower bound obtained for Maximum Likelihood Decoding and Weight Distribution is at best $r^{\Omega(k^{1/4})}$, and this only holds if the number r of input vectors is at most linear in the dimension n. Thus, their lower bound leaves open the possibility of algorithms that solve these problems in time significantly faster than testing all k subsets of r vectors. In contrast, if $r^k < 2^n$, then our $\min(r^{\Theta(k)}, 2^{\Theta(n)})$ lower bound shows one cannot do polynomially better than this testing algorithm.

Once $r^k > 2^n$, we provide a much faster algorithm based on the Fast Fourier Transform (FFT). Even when $r^k < 2^n$, we still achieve a polynomially better algorithm than the testing algorithm. In this case our algorithm's complexity is $2^{O(k)}\binom{r}{\lceil k/2 \rceil}\mathrm{poly}(n)$.

For the (k, r)-MinLinDependence$_q$ problem, our $\min(r^{\Theta(k)}, 2^{\Theta(n)})$ lower bound continues to hold provided that the characteristic of $\mathbb{F}_q$ does not divide k. Here we give a deterministic algorithm that runs in $2^{O(k^2+n)}$ time. We leave it as an open question whether this can be reduced to $2^{O(k+n)}$ time.

1.2 Techniques and Comparison to Previous Work

1.2.1 Lower Bounds

Our starting point for the lower bound is the recent $N^{\Omega(k)}$ bound for the k-SUM problem on N integers of [41]. We briefly review their proof in order to compare it to ours.

At the heart of their reduction is a way of creating integers $\zeta_1, \ldots, \zeta_N$ from partial assignments $\mathcal{A}_1, \ldots, \mathcal{A}_N$ to variables of a One-in-Three-SAT formula F, i.e., F is a formula that evaluates to true iff exactly one literal per clause evaluates to true. This is done in such a way that there is a sum $\zeta_{i_1} + \cdots + \zeta_{i_k} = M$ iff the partial assignments $\mathcal{A}_{i_1}, \ldots, \mathcal{A}_{i_k}$ corresponding to these integers can be patched together to form a consistent assignment to all variables that causes F to evaluate to true. Here, M is the positive integer that, when written in base $k+1$, equals 1^{k+c}, where c is the number of clauses. The idea is to partition the n variables arbitrarily into k equal-sized groups G_i. For each G_i, the reduction generates a new integer ζ_j for each assignment to the variables in that group. The ζ_j are interpreted in base $k+1$. The first k digits of each ζ_j are set so that the i-th digit is 1 if ζ_j comes from the i-th block, otherwise it is 0. The digit of ζ_j corresponding to a clause is 1 if the assignment of the variables corresponding to ζ_j causes exactly one of the literals of the clause to be true. In order to obtain a sum $\zeta_{i_1} + \cdots + \zeta_{i_k} = M$, one must take an integer associated with each block, so one obtains a consistent assignment, and each clause must have exactly one literal set to true.

By replacing "digits" with "coordinates" and "integers" with "vectors", the proof of [41] shows a $\min(2^{\Theta(n)}, r^{\Omega(k)})$ hardness for (k, r)-ZEROSUM$_q$ and (k, r)-TARGETSUM$_q$ with the following restrictions:

1. the characteristic of the field $\mathbb{F}_q$ must be larger than k, and
2. r belongs to a geometric sequence of numbers, rather than being an arbitrary integer.

We are not able to modify the proof of [41] to remove these restrictions. The issue is that when the characteristic is 2, there are cancellations that lead to false positives in the reduction of [41]. Indeed, trying to adapt their reduction to binary fields would instead require hardness of the problem ODD-SAT, the problem of having an odd number of literals evaluate to true in each clause. However, this latter problem is in P via Gaussian elimination.

We instead base our reduction on the NOT-ALL-EQUAL-SAT problem, the problem of having one or two out of three literals evaluate to true in each clause. We again partition the variables into blocks, but now we want a clause coordinate to be 1 iff one or two of its literals evaluates to one. The obstacle, though, is that for a given block, not all the variables associated with a clause may be assigned to that block. For instance, a clause on three variables may have its variables assigned in three different blocks. It is easy to see that some interaction between the blocks is needed to enforce consistency. By using a version of NOT-ALL-EQUAL-SAT in which each variable occurs a bounded number of times, we are able to introduce a linear number of new variables and dimensions, which overall have the effect of allowing the blocks to communicate in a way that a consistent assignment is enforced.

This approach allows us to conclude hardness for a sequence $r_0 < r_1 < r_2 < \cdots$ of values to r. We show that if there were an r between r_{i-1} and r_i for which the problem were "easy", this would contradict the hardness of the problem on r_i vectors. This does not follow from standard "padding arguments", since we must have distinct vectors and cannot create new dependences.

It is worth pointing out again that our bounds apply for the full range of r and k, in contrast to previous work.

1.2.2 Upper Bounds

Our algorithms for

- (k, r)-ZEROSUM$_q$,
- (k, r)-TARGETSUM$_q$,
- (k, r, v)-ZEROSUM$_q$,
- (k, r, v)-TARGETSUM$_q$, and
- (k, r)-LINDEPENDENCE$_q$

problems are based on the FFT and convolution.

Our algorithm for (k, r)-MINLINDEPENDENCE$_q$ is one step beyond this. We choose a small (in fact, pairwise independent) family of linear maps from $\mathbb{F}_{q^n}$ to $\mathbb{F}_{q^k}$ such that for any minimal k-dependence $x_{i_1}, \ldots, x_{i_k}$ in the input, there exists a linear map h in our family for which the image of $x_{i_1}, \ldots, x_{i_k}$ under h is a minimal k-dependence. We can find such a minimal k-dependence by testing all minimally k-dependent vectors in $\mathbb{F}_{q^k}$, each test using several applications of the FFT. The small set of functions can be chosen in a variety of ways. The overall technique of hashing followed by fast convolution bears a strong similarity to the color-coding method of Alon, Yuster and Zwick [3] which applies hashing followed by fast matrix multiplication in order to find copies of a small subgraph inside a given graph.

2 Preliminaries

The following standard claim is useful.

Claim 5 *Let $q > 2$ be a prime power and let $x_1, \ldots, x_n$ be n elements chosen independently and uniformly at random from $\mathbb{F}_q^n$, then the probability that they are linearly independent is at least $e^{-\frac{2}{q}}$. Equivalently, a random n by n matrix over $\mathbb{F}_q$ is non-singular with probability at least $e^{-\frac{2}{q}}$.*

Our upper bounds depend on efficient algorithms that compute the Fourier coefficients over any abelian group. Fast Fourier Transform (FFT) algorithms first appeared in [14] for the cyclic group $\mathbb{Z}/n\mathbb{Z}$ and later were generalized to any abelian groups (see, e.g., the survey article [37]).

Fact 6 (Fast Fourier Transform) *Let $\mathbb{F}_q$ be the finite field with q elements. Let $f : \mathbb{F}_q^n \to \mathbb{C}$ be a complex-valued function defined over $\mathbb{F}_q^n$. Then there is a Fast Fourier Transform (FFT) algorithm which compute all the Fourier coefficients of f in time $O(nq^n)$.*

Fact 7 *Let $f_1, \ldots, f_k : \mathbb{F}_q^n \to \{0, 1\}$ be k Boolean functions defined on $\mathbb{F}_q^n$, then the number of elements $(x_1, \ldots, x_k)$ such that $x_1 + \cdots + x_k = 0$ and $f_1(x_1) = \cdots f_k(x_k) = 1$ can be computed from the Fourier coeffi-*

cient of the convolution of these k functions:

$$\begin{aligned}
&|\{\langle x_1, \ldots, x_k \rangle : x_1 + \cdots + x_k = 0 \\
&\textit{and } f_i(x_i) = 1 \textit{ for each } i = 1, \ldots, k \}| \\
&= q^{n(k-1)}(f_1 * f_2 * \cdots * f_k)(0) \\
&= q^{n(k-1)} \sum_{\alpha \in \mathbb{F}_q^n} \prod_{j=1}^{k} \hat{f}_j(\alpha). \qquad (1)
\end{aligned}$$

3 Hardness

Theorem 8 *Given function $k : \mathbb{Z}^+ \to \mathbb{Z}^+$ such that $k(n) < n$ for all $n \in \mathbb{Z}^+$ and function $r : \mathbb{Z}^+ \to \mathbb{Z}^+$ such that $k(n) < r(n) < q^n$ for all $n \in \mathbb{Z}^+$, then the (k, r)-TARGETSUM$_q$ problem requires at least $\min\left(\binom{r(n)}{\beta k(n)}, 2^{\beta n}\right)$ time for some constant $\beta < 1$, unless d-SAT on n variables can be solved in $2^{O(dn)\beta^{1/O(d)}}$ time for any $d \geq 3$.*

Proof Suppose q is a power of some prime $p \geq 2$. Let F be an instance of d-SAT with n variables and m clauses. For some $\epsilon > 0$ to be specified later, we use the improved Sparsification Lemma of Calabro, Impagliazzo and Paturi [9] to reduce F to a collection of $2^{\epsilon n}$ d-SAT formulas, with each formula having n variables and $n \cdot (d/\epsilon)^{O(d)}$ clauses. Next, we use a standard reduction to convert each d-SAT formula to a 3-SAT formula with $f \stackrel{\text{def}}{=} O(nd)(d/\epsilon)^{O(d)}$ variables and clauses.

Now, we convert each 3-SAT formula to an NAE-SAT formula by a standard reduction: each clause $(v_1 \vee v_2 \vee v_3)$ is replaced by three clauses $(v_1 \vee v_2 \vee x) \wedge (\neg x \vee v_3 \vee y) \wedge (x \vee y \vee \alpha)$ where x and y are new variables and α is a common variable used across the clauses in the formula. Furthermore, we can ensure that each variable occurs only a constant number of times in each NAE-SAT formula by replacing duplicate copies of a variable by distinct variables and introducing equality constraints (two variables x and y can be constrained to be equal by two NAE-SAT clauses $(\neg x \vee y) \wedge (x \vee \neg y)$). The number of variables and clauses in each formula remains $O(f)$.

Next, we reduce each NAE-SAT formula to a separate (k, r_k)-TARGETSUM$_q$ problem, where the function $r_k : \mathbb{Z}^+ \to \mathbb{Z}^+$ will be specified later. Fix an arbitrary ordering of the literals inside each clause of the formula. For any literal ℓ, let us denote by $v(\ell)$ the variable corresponding to the literal. To start off the reduction, for each clause $(\ell_1 \vee \ell_2 \vee \ell_3)$ in the NAE-SAT formula, where ℓ_1, ℓ_2, ℓ_3 are literals, we introduce three, possibly new, variables $(v(\ell_1), v(\ell_2))$, $(v(\ell_2), v(\ell_3))$ and $(v(\ell_3), v(\ell_1))$. We call each such variable (a, b) a *pairvar*. The number of pairvars is at most three times the number of clauses, $O(f)$. Next, for a function $k' : \mathbb{Z}^+ \to \mathbb{Z}^+$ to be specified later, partition the original set of variables arbitrarily into $k' = k'(n)$ *blocks*, of sizes within a constant factor of each other, and assign an arbitrary ordering among the blocks. For each pairvar (a, b), if both a and b belong to the same block, we include the pairvar in that block. Otherwise, we include it in the first block containing either a or b. Thus, each variable (original or pairvar) is contained in exactly one block. Also, since each variable occurs a constant number of times, each block contains $O(f/k')$ original and pairvar variables.

We now generate the (k, r_k)-TARGETSUM$_q$ instance. Each block will yield $2^{O(f/k')}$ many elements of $\mathbb{F}_2^{n'}$, where n' will be $O(f)$. Consider the i'th block, with $i \in [k']$. Let A_i be the set of all possible $0/1$-assignments to the set of variables $\{x \mid x \text{ is an original variable in block } i\} \cup \{a \mid \exists \text{ pairvar } (a, b) \text{ or } (b, a), \text{not necessarily in block } i$, with b in block $i\}$. Note that an assignment in A_i fixes the values of all pairvars in block i. For each assignment $\alpha \in A_i$, we produce an element $x_\alpha \in \mathbb{F}_2^{n'}$ in the following way. The first k' bits of x_α are 0, except for the i'th which is 1. Next, there is a coordinate for each clause C in the formula, called the *clause coordinate*. If $C = (\ell_1 \vee \ell_2 \vee \ell_3)$, its clause coordinate value is set to:

$$\sum_{i \in [3]:\ v(\ell_i) \text{ in block } i} \alpha(\ell_i) \bmod p \\ - \sum_{(i,j) \in [3]^2 : (v(\ell_i), v(\ell_j)) \text{ in block } i} \alpha(\ell_i)\alpha(\ell_j) \bmod p.$$

The rest of the coordinates, called the *consistency coordinates*, will be set so as to ensure consistency among assignments to the pairvars by different blocks. We partition the consistency coordinates into pairs and index each pair with a pairvar. For the pair of coordinates indexed by pairvar (a, b), if neither a nor b is in block i, then both these coordinates are set to 0, and the same if both a and b are in block i. Otherwise, if a is in block i but b is not, then the first coordinate is set to $\alpha(a)$ and the second to $-\alpha(b)$, and similarly, if a is not in block i but b is, then the first coordinate is set to $-\alpha(a)$ and the second to $\alpha(b)$. This completes the description of x_α. The target vector z for the (k, r)-TARGETSUM$_q$ instance is set to $1^{n'-2p} \circ 0^{2p}$ where p is the total number of pairvars. To make n' independent of k', we can pad all the strings with extra zeroes at the end.

In the above construction, we define functions k' and r_k such that $k'(n) = k(n')$ and $r_k(n') = \sum_{i \in [k'(n)]} |A_i|$ for every $n \geq 1$, where n' and the A_i's are as above. Thus, we obtain a valid (k, r_k)-TARGETSUM$_q$ instance, where

$r_k(n) = k(n) \cdot 2^{O(n/k(n))}$. To see the correctness of the reduction, suppose there are $x_{\alpha_1}, \ldots, x_{\alpha_{k'}}$ such that $x_{\alpha_1} + \cdots + x_{\alpha_{k'}} = z$. First, assume that all the pairvars are assigned consistently by the assignments $\alpha_1, \ldots, \alpha_{k'}$. Because each x_{α_i} has a 1 in only one of the first k' coordinates, and z has 1's on all the first k' coordinates, each α_i is in A_i without loss of generality. Since consistency of the pairvars assignments is assumed, the partial assignments α_i can be combined to obtain an assignment α to all the original variables. The claim is that α is a satisfying assignment to the NAE-SAT formula. Take a clause $C = (\ell_1 \vee \ell_2 \vee \ell_3)$ from the NAE-SAT formula. If we add up, modulo p, the value of the clause coordinate corresponding to C for each x_{α_i}, then this sum must equal:

$$\begin{aligned} S_C &= (\alpha(\ell_1) + \alpha(\ell_2) + \alpha(\ell_3) - \alpha(\ell_1)\alpha(\ell_2) \\ &\quad -\alpha(\ell_2)\alpha(\ell_3) - \alpha(\ell_3)\alpha(\ell_1)) \bmod p \end{aligned}$$

- If three literals in C are assigned 1, then $S_C = 1 + 1 + 1 - 1 - 1 - 1 = 0$.
- If two literals in C are assigned 1, then $S_C = 1 + 1 + 0 - 1 - 0 - 0 = 1$.
- If one literal in C is assigned 1, then $S_C = 1 + 0 + 0 - 0 - 0 - 0 = 1$.
- If no literal in C is assigned 1, then $S_C = 0 + 0 + 0 - 0 - 0 - 0 = 0$.

Since all the clause coordinates of z are set to 1, it must be the case that α satisfies the NAE-SAT formula.

It remains to check that the pairvars are set consistently. For the pair of consistency coordinates indexed by a pairvar (a, b), either these coordinates are zero in all of the x_{α_i}'s, or there exist $i \neq j$ such that these coordinates are nonzero in x_{α_i} and x_{α_j} but they are zero for all the other strings. In the first case, there is no consistency issue. The second case occurs when one of a and b is in block i and the other is in block j. But then, because the value of $x_{\alpha_i} + x_{\alpha_j}$ is zero at the consistency coordinates indexed by (a, b), it must be the case that $\alpha_i(a) - \alpha_j(a) = 0$ and $\alpha_i(b) - \alpha_j(b) = 0$.

Thus, the reduction yields $2^{\epsilon n}$ (k, r_k)-TargetSum$_q$ instances on $n' = O(dn)(d/\epsilon)^{O(d)}$ many coordinates, with $r_k(n') = k' \cdot 2^{O(f/k')} = k' \cdot 2^{nd(d/\epsilon)^{O(d)}/k'} = k' \cdot 2^{nd/(k'\sqrt{\delta})}$ where the last equality follows by choosing $\epsilon = d\delta^{1/\gamma d}$ for an appropriate value of γ. Therefore, if the output of the reduction can be solved in time $\binom{r_k(n')}{\delta k(n')}$, then, using the standard bound $\binom{a}{b} \leq (ae/b)^b$, an arbitrary d-SAT on n variables can be solved in time $2^{\epsilon n} \cdot (e/\delta)^k 2^{nd\sqrt{\delta}} = 2^{O(dn)\delta^{1/O(d)}}$.

We need to show how to reduce a (k, r_k)-TargetSum$_q$ to a (k, r)-TargetSum$_q$ instance for an arbitrary function $r : \mathbb{Z}^+ \to \mathbb{Z}^+$. First consider the case of $r(n) \leq r_k(n)$. For $i \in [\lceil n/k(n) \rceil]$, let $k_i : \mathbb{Z}^+ \to \mathbb{Z}^+$ be defined so that $k_i(n) = n/i$. Note that $k_i(n) < n$ if $k(n) < n$ for all positive i. Now, apply the reduction above to get an instance of (k_i, r_{k_i})-TargetSum$_q$ of size $ik(n)$ that requires $\binom{r_{k_i}(ik(n))}{\delta k_i(ik(n))}$ time, unless d-SAT on n variables can be solved in $2^{O(dn)\delta^{1/O(d)}}$ time. We can pad the strings of such an instance with zeroes in order to get an instance of (k, r_i)-TargetSum$_q$ of size n with the same hardness guarantee, where $r_i(n) = r_{k_i}(ik(n)) = k(n) \cdot 2^{O(i)}$. Now, for the given r, suppose $r_i(n) < r(n) < r_{i+1}(n)$ for some $i \in [\lceil n/k(n) \rceil - 1]$. We show how to reduce (k, r_{i+1})-TargetSum$_q$ to (k, r)-TargetSum$_q$. We need the following claim.

Claim 9 *For positive integers $k < m < n$, there exists a collection $\mathcal{C}$ of subsets of $[n]$ such that each subset $S \in \mathcal{C}$ is of size m and for any subset $I \subset [n]$ of size k, there exists $S \in \mathcal{C}$ containing I. The size of $\mathcal{C}$ is at most $(12n/m)^k$, and it can be constructed deterministically in the same amount of time.*

Proof Arbitrarily partition $[n]$ into nearly equal-sized buckets, each of size either $\lceil m/2k \rceil$ or $\lfloor m/2k \rfloor$. The number of buckets is at most $4nk/m$. Consider all subsets of exactly k of these buckets. There are $\binom{4nk/m}{k} \leq (4en/m)^k$ many such subsets. For each choice of k buckets, take the union S of items in these buckets. S contains at most $m/2 < m$ items; add $m - |S|$ many arbitrary distinct additional items to S so as to make the size of S equal to m. The collection of all S satisfies our claim. ■

Apply Claim 9 with k as above, $m = r$, $n = r_{i+1}$. The size of the collection $\mathcal{C}$ we get is $2^{O(k)}$. Now, suppose there is an algorithm A_r for (k, r)-TargetSum$_q$. Given $x_1, \ldots, x_{r_{i+1}}$ and a target vector z, for every $S \in \mathcal{C}$, run A_r with input $\{x_i : i \in S\}$ and the same target vector z. If indeed there exists a solution of (k, r_{i+1})-TargetSum$_q$, A_r should accept for some choice of $S \in \mathcal{C}$. Hence, if A_r runs in time $2^{-O(k)}\binom{r}{\delta k} = \binom{r}{O(\delta)k}$, then (k, r_{i+1})-TargetSum$_q$ can be solved in time $\binom{r}{\delta k} \leq \binom{r_{i+1}}{\delta k}$, implying there is an algorithm for d-SAT running in time $2^{O(dn)\delta^{1/O(d)}}$.

It remains to consider the case of $r(n) > r_k(n)$. Define $\ell : \mathbb{Z}^+ \to \mathbb{Z}^+$ so that $r(n) - k(n) = r_\ell(n - k(n))$ for every $n \geq 1$. First, obtain a hard instance of (ℓ, r_ℓ)-TargetSum$_q$ of size $n - k(n)$ by the earlier reduction. The instance consists of $r(n) - k(n)$ vectors $x_1, \ldots, x_{r-k} \in \mathbb{F}_q^{n-k}$ and a target vector $z \in \mathbb{F}_q^{n-k}$, and say $x_1, \ldots, x_s$ for some $s < r - k$ consists of the vectors that arise out of assignments to the first block in the reduction from d-SAT. We construct an instance of (k,r)-TARGETSUM$_q$ of size n, consisting of $y_1,\ldots,y_r \in \mathbb{F}_q^n$ and target vector $w \in \mathbb{F}_q^n$. Set w to $z \circ 0^k$. For $i \in [k]$, set y_{r-k+i} to $0^{n-k} \circ e_i$ where $e_i \in \mathbb{F}_2^k$ has all 0's except for 1 at the ith position. For $i \in [s+1, r-k]$, set $y_i = x_i \circ 0^k$. Finally, for $i \in [s]$, set $y_i = x_i \circ v$ where $v \in \mathbb{F}_2^k$ is the vector with -1's in the first $k-\ell$ positions and 0's in the rest. Observe that any solution to this (k,r)-TARGETSUM$_q$ instance, when restricted to the first $r-k$ coordinates, must yield a solution to the (ℓ, r_ℓ)-TARGETSUM$_q$ instance, and so in particular, must contain one of $y_1,\ldots,y_s$. But then, to satisfy the constraints on the last k coordinates, the solution must also contain $\{y_{r-k+1},\ldots,y_{r-\ell}\}$. This gives a correspondence between solutions to the (ℓ, r_ℓ)-TARGETSUM$_q$ instance and the (k,r)-TARGETSUM$_q$ instance. Hence, unless there is an algorithm to solve d-SAT in $2^{O(dn)\delta^{1/O(d)}}$ time, solving (r,k)-TARGETSUM$_q$ requires at least $\binom{r_\ell(n/2)}{\delta\ell(n/2)} = 2^{\beta n}$ time for some constant $\beta < 1$. ■

Now, consider the (k,r,v)-TARGETSUM$_q$ problem, where k and r are as in Theorem 8 and v denotes an arbitrary family of vectors in $(\mathbb{F}_q \backslash \{0\})^n$. We observe that the above proof of Theorem 8 also shows hardness for this problem. Specifically, in the reduction from NAE-SAT, multiply each vector arising from block i by v_i^{-1}, for every $i \in [k'(n)]$. It is easy to see that this gives a reduction from NAE-SAT to (k, r_k, v)-TARGETSUM$_q$ for the same function r_k as in the above proof. The rest of the proof goes through straightforwardly, with the only other nontrivial modification occuring in the last paragraph of the proof where we again need to multiply the vectors being appended by the appropriate scaling factors. We have thus proved the following theorem.

Theorem 10 *Given $k, r : \mathbb{Z}^+ \to \mathbb{Z}^+$ as in Theorem 8 and an arbitrary family of vectors $v = (v_1,\ldots,v_{k(n)}) \in (\mathbb{F}_q \backslash \{0\})^{k(n)}$ for every $n \geq 1$, the (k,r,v)-TARGETSUM$_q$ problem requires at least $\min\left(\binom{r(n)}{\beta k(n)}, 2^{\beta n}\right)$ time for some constant $\beta < 1$, unless d-SAT on n variables can be solved in $2^{O(dn)\beta^{1/O(d)}}$ time for any $d \geq 3$.*

For the (k,r,v)-ZEROSUM$_q$ problem, we have already observed that the problem is trivial if $s_v = \sum_{i=1}^{k(n)} v_i = 0$ over $\mathbb{F}_q$. But if $s_v \neq 0$, then we can again show the same hardness as above by reducing from (k,r,v)-TARGETSUM$_q$. Given $x_1,\ldots,x_r$ and target vector z, define $y_i = x_i - s_v^{-1} z$ for every $i \in [r]$. Now, if the instance of (k,r,v)-ZEROSUM$_q$ with inputs $y_1,\ldots,y_r$ has a solution $i_i,\ldots,i_k \in [r]$ such that $\sum_{i=1}^k v_i y_i = 0$, then $\sum_{i=1}^k v_i x_i = z$ and vice versa. Therefore:

Theorem 11 *Given $k, r : \mathbb{Z}^+ \to \mathbb{Z}^+$ as in Theorem 8 and a family of vectors $v = (v_1,\ldots,v_{k(n)}) \in (\mathbb{F}_q \backslash \{0\})^{k(n)}$ such that $\sum_{i=1}^{k(n)} v_i \neq 0$ for every $n \geq 1$, the (k,r,v)-ZEROSUM$_q$ problem requires at least $\min\left(\binom{r(n)}{\beta k(n)}, 2^{\beta n}\right)$ time for some constant $\beta < 1$, unless d-SAT on n variables can be solved in $2^{O(dn)\beta^{1/O(d)}}$ time for any $d \geq 3$. Specifically, this hardness holds for the (k,r)-ZEROSUM$_q$ problem if $k(n) \neq 0 \pmod p$ where p is the characteristic of $\mathbb{F}_q$.*

For the (k,r)-LINDEPENDENCE$_q$ problem also, we can show the same hardness, this time by examining the proof of Theorem 8. Observe that for the output of the reduction from the NAE-SAT instance in the proof, not only is the generated target vector the sum of k' vectors iff the NAE-SAT formula is satisfiable but actually, the generated target vector is in the span of k' vectors iff the NAE-SAT formula is satisfiable. The rest of the proof goes through straightforwardly.

Theorem 12 *Given $k, r : \mathbb{Z}^+ \to \mathbb{Z}^+$ as in Theorem 8, the (k,r)-LINDEPENDENCE$_q$ problem requires at least $\min\left(\binom{r(n)}{\beta k(n)}, 2^{\beta n}\right)$ time for some constant $\beta < 1$, unless d-SAT on n variables can be solved in $2^{O(dn)\beta^{1/O(d)}}$ time for any $d \geq 3$.*

4 Algorithms for (k,r,v)-TARGETSUM$_q$ and (k,r)-LINDEPENDENCE$_q$

We show how to solve the (k,r,v)-TARGETSUM$_q$ problem in $2^{O(k)}\binom{r}{\lceil k/2 \rceil} \cdot \mathrm{poly}(n)$ time for constant q, which improves the $\binom{r}{k} \cdot \mathrm{poly}(n)$ time algorithm of exhaustive search. This implies a solution for (k,r)-TARGETSUM$_q$, (k,r,v)-ZEROSUM$_q$, and (k,r)-ZEROSUM$_q$. By enumerating over different v, it can also be used to solve (k,r)-LINDEPENDENCE$_q$ with a blowup of an additional q^k factor.

The basic idea is most easily seen for $q = 2$ and an even integer k. In this case $v = 1^k$. We form a table T of all possible vector sums of $k/2$ vectors $x_{i_1},\ldots,x_{i_{k/2}}$ from $x_1, x_2, \ldots, x_r$. Next, for each vector $w \in T$, we check if $w \oplus z \in T$, where $z \in F_2^n$ is the target vector. Since we do not require that $i_1, i_2, \ldots, i_k$ are distinct, if there is a sum of k input vectors that equals z, we will find it, and if we find such a sum, it solves the (k,r,v)-TARGETSUM$_2$ problem. The time is clearly $\binom{r}{k/2} \cdot \mathrm{poly}(n)$. Handling odd k is straightforward - we can build a table T_1 of all sums of $\lceil k/2 \rceil$ input vectors, and a table T_2 of all sums of $\lfloor k/2 \rfloor$ input vectors, and check if there is a vector $w \in T_1$ for which $w \oplus z \in T_2$, which can be done in $(|T_1| + |T_2|)\mathrm{poly}(n)$ time.

We now descibe the extension for general q. The main difference is that we first group the coefficients of $v = (v_1, \ldots, v_k)$ based on their value in $\mathbb{F}_q \setminus \{0\}$. Let $z \in \mathbb{F}_q^n$ be the target vector. Let the input vectors be $x_1, \ldots, x_r \in \mathbb{F}_q^n$, so we want to find a sub-multiset $x_{i_1}, \ldots, x_{i_k}$ for which $\sum_{j=1}^k v_j x_{i_j} = z$.

In the first stage, for each subset A of the input vectors of size $\lceil k/2 \rceil$, the algorithm considers a special set of $2^{O(k)}$ sequences $(x_{i_1}, \ldots, x_{i_{\lceil k/2 \rceil}})$ of $\lceil k/2 \rceil$ elements of A with repetition. Here, some elements of A may not be included in a given sequence in the set.

The special set of sequences for the subset A is formed as follows. For a sequence $(x_{i_1}, \ldots, x_{i_{\lceil k/2 \rceil}})$ of $\lceil k/2 \rceil$ elements of A (with repetition), for each $\ell \in \mathbb{F}_q \setminus \{0\}$, let B_ℓ be the multiset of x_{i_j} for which $v_j = \ell$. Note that $|B_\ell|$ is the number of coordinates of v which equal ℓ. Let C_ℓ be the multiset obtained from B_ℓ by taking each distinct element of B_ℓ, and reducing its multiplicity modulo the characteristic of $\mathbb{F}_q$. For any two sequences $(x_{i_1}, \ldots, x_{i_{\lceil k/2 \rceil}})$ and $(y_{i_1}, \ldots, y_{i_{\lceil k/2 \rceil}})$ which result in the same collection of multisets $\{C_\ell\}$, we have $\sum_{j=1}^{\lceil k/2 \rceil} v_j x_{i_j} = \sum_{j=1}^{\lceil k/2 \rceil} v_j y_{i_j}$. In this case we say sequences $(x_{i_1}, \ldots, x_{i_{\lceil k/2 \rceil}})$ and $(y_{i_1}, \ldots, y_{i_{\lceil k/2 \rceil}})$ are *equivalent*.

The special set of sequences we use for the subset A is a maximal set of non-equivalent sequences, and we call such a set of sequences a *representative sequence set*. The number of sequences in the representative sequence set is bounded by $2^{O(k)}$ for constant q. To see this, for each distinct element x of A, and for each of the at most $q-1$ different C_ℓ, we choose a number between 0 and the characteristic of $\mathbb{F}_q$, minus one, which is at most q, representing the number of occurrences of x in C_ℓ. Since A has at most $\lceil k/2 \rceil$ distinct elements, the number of choices we have is $q^{O(qk)} = 2^{O(k)}$ for constant q. However, not all such collections of $\{C_\ell\}$ correspond to a sequence of exactly $\lceil k/2 \rceil$ elements of A. For each such collection of $\{C_\ell\}$, there is such a sequence if and only if $|B_\ell| - |C_\ell|$ is a multiple of the characteristic of $\mathbb{F}_q$ for all ℓ. Indeed, in this case, case we can arbitrarily increase the multiplicity of a vector in C_ℓ by a multiple of the characteristic. On the other hand, any sequence gives rise to a collection $\{C_\ell\}$ with the property that $|B_\ell| - |C_\ell|$ is a multiple of the characteristic of $\mathbb{F}_q$ for all ℓ. For each sequence in the representative sequence set, the algorithm first computes the vector $\sum_{j=1}^{\lceil k/2 \rceil} v_j x_{i_j}$. This can be done in $2^{O(k)} \binom{r}{\lceil k/2 \rceil}$ time.

In the second stage, for each sub-multiset B of the input vectors of size $\lfloor k/2 \rfloor$, the algorithm considers all sequences $(x_{i_1}, \ldots, x_{i_{\lfloor k/2 \rfloor}})$ of B from a representative sequence set, and computes $\sum_{j=1}^{\lfloor k/2 \rfloor} v_{j+\lceil k/2 \rceil} x_{i_j}$.

Then there is a solution to the (k, q, v)-TARGETSUM problem if and only if there is a vector w computed in the first stage for which the vector $-w + z$ is computed in the second stage. This can be easily tested in $2^{O(k)} \binom{r}{\lceil k/2 \rceil} \cdot \text{poly}(n)$ time. We thus have:

Theorem 13 *(k, r, v)-TARGETSUM$_q$ can be solved in deterministic $2^{O(k)} \binom{r}{\lceil k/2 \rceil} \cdot \text{poly}(n)$ time.*

When $r = 2^{\Omega(n/k)}$, we can do better and match the lower bound from Theorem 8, up to a constant factor in the exponent. Again, suppose we have an instance of (k, r, v)-TARGETSUM$_q$ with inputs $x_1, \ldots, x_r \in \mathbb{F}_q^n$ and target vector $z \in \mathbb{F}_q^n$. For $i \in [k]$, define the set $S_i \subseteq \mathbb{F}_q^n$ to be $\{v_i x_j \mid j \in [r]\}$, and let $f_i : \mathbb{F}_q^n \to \{0, 1\}$ be the indicator function of S_i. Now, Fact 7, concerning convlution and the FFT, directly leads to:

Theorem 14 *(k, r, v)-TARGETSUM$_q$ can be solved in deterministic $O(n \cdot \log k(n) \cdot q^n)$ time.*

5 (k, r)-MINLINDEPENDENCE$_q$ Algorithms and Algorithms for Related Problems

5.1 An algorithm for the decision problem

In this section, we show an algorithm for the (k, r)-MINLINDEPENDENCE$_q$ problem which is tight for $k(n) = O(\sqrt{n})$ but is not for larger k.

Theorem 15 *For functions $k, r : \mathbb{Z}^+ \to \mathbb{Z}^+$, the (k, r)-MINLINDEPENDENCE$_q$ problem can be solved in $O\left(\text{poly}(n) \cdot \min\left(\binom{r(n)}{k(n)}, q^{O(n+k^2(n))}\right)\right)$ deterministic time.*

Proof The algorithm with running time $\text{poly}(n)\binom{r(n)}{k(n)}$ simply picks each $k(n)$-sized subset of the $r(n)$ inputs and checks if they are minimally dependent using Gaussian elimination. For the other upper bound, we first give a randomized algorithm which is easy to describe, and for which there is a standard way to derandomize it.

Choose uniformly at random a full-rank linear map $L : \mathbb{F}_q^n \to \mathbb{F}_q^{k-1}$. The algorithm passes if and only if there are $x_{i_1}, \ldots, x_{i_k}$ for $i_1, \ldots, i_k \in [r]$ such that: (i) $L(x_{i_1}), \ldots, L(x_{i_{k-1}})$ are linearly independent, and (ii) $x_{i_k} \in \text{span}(x_{i_1}, \ldots, x_{i_{k-1}})$. We need to justify the success probability of this algorithm as well as its running time.

Suppose there is no subset of k minimally dependent elements. In this case, note that if $L(x_{i_1}), \ldots, L(x_{i_{k-1}})$

are linearly independent, then $x_{i_1}, \ldots, x_{i_{k-1}}$ are also linearly independent. Therefore, the above algorithm will fail with probability 1. On the other hand, suppose that the input contains a set of k minimally dependent elements $x_{i_1}, \ldots, x_{i_k}$. Then, $x_{i_1}, \ldots, x_{i_{k-1}}$ are linearly independent. Therefore, the probability that $L(x_{i_1}), \ldots, L(x_{i_{k-1}})$ are linearly independent is exactly equal to the probability that $k-1$ elements, uniformly chosen from $\mathbb{F}_q^{k-1}$, are linearly independent. This probability is lower bounded by a constant by Claim 5, and so, the algorithm passes with constant probability. Finally, note that since the given algorithm is one-sided, we can amplify the success probability to any required threshold by running it $O(1)$ times and passing if any of the runs passes.

We now describe how to get the claimed running time. We repeat the following for each choice of $v_1, \ldots, v_{k-1} \in \mathbb{F}_q$ that are not all zero. For each choice of $k-1$ linearly independent elements $y_{i_1}, \ldots, y_{i_{k-1}} \in \mathbb{F}_q^{k-1}$, with y_k defined as $v_1 y_{i_1} + \cdots + v_k y_{i_{k-1}}$, we will show how to efficiently check if there exist any $x_{i_1}, \ldots, x_{i_k}$ such that $L(x_{i_1}) = y_{i_1}, \ldots, L(x_{i_k}) = y_{i_k}$ and $x_1 + \cdots + x_{k+1} = 0$. For each of the at most $\binom{q^k}{k} \leq q^{O(k^2)}$ choices of $y_1, \ldots, y_k$, we will achieve this in $\tilde{O}(q^n \text{poly}(n))$ time, proving our claim. So, fix linearly independent $y_{i_1}, \ldots, y_{i_{k-1}} \in \mathbb{F}_q^k$ and set y_{i_k} to $v_1 y_{i_1} + \cdots + v_{k-1} y_{i_{k-1}}$. Let H equal $\mathsf{Ker}(L) = \{x : L(x) = 0\}$; H is a subspace of dimension $n-k+1$. For each $j \in [k]$, we have that $L^{-1}(y_{i_j})$ is a coset $v_j + H$. For each $j \in [k]$, we define $f_j : H \to \{0,1\}$ as $f_j(x) = \mathbb{I}(v_j + x)$, where $\mathbb{I}(x) = 1$ if x is one of the r inputs and 0 otherwise. Now, observe that there exist $x_{i_1} \in L^{-1}(y_{i_1} 1), \ldots, x_{i_k} \in L^{-1}(y_{i_k})$ such that $x_{i_1} + \cdots + x_{i_k} = 0$ if and only if $(f_1 * f_2 * \cdots * f_k)(0) > 0$. By Fact 7, concerning convlution and the FFT, we can compute the convolution in $O(nq^n)$ time, proving the running time bound.

Derandomization. We can choose a family $\mathcal{H}$ of pairwise-independent hash functions from $\mathbb{F}_{q^n}$ to $\mathbb{F}_{q^{3k}}$ as follows. We choose $a \in \mathbb{F}_{q^n}$ and $b \in \mathbb{F}_{q^{3k}}$ arbitrarily, and our map is $[a \cdot x]_k + b$, where $[y]_k$ denotes the restriction to the last k coordinates of y. Then $|\mathcal{H}| = q^{n+3k}$. Such a family is known to be pairwise-independent. Although the family is not linear, it is affine, and we know the offset b, so we can perform the above algorithm by looking for sets of k vectors in the range with the property that any non-trivial linear combination of them that spans a scalar multiple of the offset b must have a non-zero coefficient multiplying every vector, and there is such a linear combination.

Suppose that S is a set of k items that forms a linear dependence that is not minimal. Then there is a linear dependence on a proper non-empty subset of these items. It follows that a multiple of b is in the span of this subset, and so it cannot map to a set of k vectors in the range that we consider.

Suppose that S is a set of k items that is linearly independent. This set will not be found by the FFT verification, even if it passes our criterion (e.g., if we look at its image).

Suppose that S is a minimal k-dependence. With probability $1 - k^2/q^{3k}$, the k vectors map to distinct images. Moreover, since the map is affine, a linear combination involving all k such vectors equals a multiple of b. It remains to check that any proper non-empty subset T of S does not span a multiple of b. There are at most q^k elements in the union of such sets T, and none can be zero since S is a minimal k-dependence. For any fixed element y, the probability that $[a \cdot y]_k$ is a multiple of b is at most q/q^{3k}, and so by a union bound none of these span a multiple of b with probability at least $1 - q^{k+1}/q^{3k}$. Hence, by a union bound S will pass the criteria of our procedure with probability at least $1 - k^2/q^{3k} - q^{k+1}/q^{3k}$. ■

5.2 An approximation algorithm for counting the number of witnesses

Note that the algorithms in Section 4 not only detect solutions to the TARGETSUM problem but also count them. It is easy to see this is the case for the first algorithm. For the second FFT-based algorithm, the output of the convolution itself gives the count. The situation is more complicated for the FFT-based algorithm for the MINLINDEPENDENCE problem. Here, we are only able to find an approximation to the total number of solutions.

Theorem 16 *For any $\epsilon > 0$, there exists a randomized algorithm that, with probability at least $2/3$, approximates the number of solutions to (k,r)-MINLINDEPENDENCE$_q$ to within a multiplicative factor of $(1 \pm \epsilon)$. The running time of the algorithm is $\tilde{O}\left(q^{O(n+k^2)}\text{poly}(n)/\epsilon^2\right)$.*

Proof The algorithm for approximate counting is essentially the same as the algorithm for detecting! As before, choose a random full-rank linear map $L : \mathbb{F}_q^n \to \mathbb{F}_q^k$. Suppose there are s solutions to (k,r)-MINLINDEPENDENCE$_q$. By Claim 5, for each such solution $x_{i_1}, \ldots, x_{i_k}$, the probability that $L(x_{i_1}), \ldots, L(x_{i_{k-1}})$ are linearly independent is at least a constant, say, p_k, and so, the expected number of solutions with linearly independent $L(x_{i_1}), \ldots, L(x_{i_{k-1}})$ is sp_k. We want to bound the concentration around this mean.

Formally, let $\mathcal{C}$ denote the set consisting of all the s solutions. For a given $c \in \mathcal{C}$, let χ_c be the indicator variable for the event that L maps $k-1$ of the elements in c to linearly independent elements, and let $X = \sum_{c \in C} \chi_c$. So, $\mathbb{E}[\chi_c] = p_k$ and $\mathbb{E}[X] = sp_k$. Also, $\mathrm{Var}[X] = \sum_{c \in \mathcal{C}} \mathrm{Var}[\chi_c] + \sum_{c \neq c' \in \mathcal{C}} \mathrm{Cov}(\chi_c, \chi_{c'})$. But note that:

$$\mathrm{Var}[\chi_c] = p_k(1-p_k) \leq p_k$$

and

$$\begin{aligned} \mathrm{Cov}(\chi_c, x_{c'}) &= \mathbb{E}[\chi_c x_{c'}] - \mathbb{E}[\chi_c]\mathbb{E}[x_{c'}] \\ &\leq \mathbb{E}[\chi_c x_{c'}] \\ &\leq p_k \end{aligned}$$

So, $\mathrm{Var}[X] \leq sp_k + s(s-1)p_k = s^2 p_k$.

Now, suppose we independently choose s full-rank linear maps $L_1, \ldots, L_s : \mathbb{F}_2^n \to \mathbb{F}_2^k$, and let Y be the average of the s independent copies of X. Then, $\mathbb{E}[Y] = \mathbb{E}[X] = sp_k$, while $\mathrm{Var}[Y] = \mathrm{Var}[X]/s \leq s^2 p_k/s$. By Chebyshev:

$$\Pr[|Y - sp_k| > \epsilon sp_k] \leq \frac{\mathrm{Var}[Y]}{\epsilon^2 s^2 p_k^2} \leq \frac{1}{\epsilon^2 p_k s}$$

Thus, choosing s to be $O(1/\epsilon^2)$ suffices to make the probability of error less than $2/3$.

The algorithm is therefore to independently choose $m = O(1/\epsilon^2)$ many full-rank linear maps $L_1, \ldots, L_m : \mathbb{F}_q^n \to \mathbb{F}_q^k$, and for each L_j, compute X_j, the number of linearly dependent elements $x_{i_1}, \ldots, x_{i_k}$ such that $L_j(x_{i_1}), \ldots, L_j(x_{i_{k-1}})$ are linearly independent. X_j is simply the appropriate scaling of the value of the computed convolution. This makes the running time $O(\mathrm{poly}(n) q^{O(n+k^2)}/\epsilon^2)$. Finally, we output $\frac{X_1 + \cdots + X_m}{sp_k}$. ■

We also note that one can find (not just decide the existence of) a witness for each of the problems we have considered without paying any asymptotic overhead on the decision algorithms. Namely, we can use self-reducibility to fix the elements $x_{i_1}, \ldots, x_{i_{k(n)}}$ sequentially and after each fixing, check if the restricted problem still has a solution. We pay an extra $O(r(n)k(n))$ for this, which is asymptotically negligible.

6 Conclusion

In this work we studied a wide range of problems that test if linear combinations of certain-sized subsets of the input vectors equal a desired target vector. For many of these problems, we resolved the time complexity (up to polynomial factors) under the Exponential Time Hypothesis. As these problems are natural extensions of the well-studied k-SUM problem, we believe they will lead to a deeper understanding of the complexity of other problems.

One question that remains open is the complexity of the problem (k,r)-ZeroLinDependence$_q$, which for functions $k, r : \mathbb{Z}^+ \to \mathbb{Z}^+$, takes as input r elements $x_1, \ldots, x_{r(n)} \in \mathbb{F}_q^n$ and checks if there exist $x_{i_1}, \ldots, x_{i_{k(n)}}$ such that 0^n can be written as a non-trivial linear combination of $x_{i_1}, \ldots, x_{i_{k(n)}}$. This is similar to the (k,r)-LinDependence$_q$ problem, but the target vector $z = 0^n$ is fixed. It is also similar to the (k,r)-ZeroSum$_2$ problem, but we require the k vectors chosen be distinct. This problem is equivalent to the parameterized complexity of testing if the minimum distance of a linear code is at most k. Indeed, if A is the $n \times r$ matrix whose columns correspond to the input vectors, then the code $\{x \mid Ax = 0\}$ has minimum distance at most k iff there is a positive answer to the (k,r)-ZeroLinDependence$_q$ problem. The parameterized complexity, as a function of r and k, of this problem is listed as an open question in [10] (see the listing under the EvenSet problem). One property is that by Corollary 3.17 of [45], if $r > 2^{cn/k}$ for a large enough constant $c > 0$, then there is always a solution to the (k,r)-ZeroLinDependence$_q$ problem.

Acknowledgments

We thank the anonymous reviewers for their useful feedback. The fourth author was supported in part by NSF Award CCR-0514771, 0728645 and 0732334.

References

[1] Noga Alon, Zvi Galil, Oded Margalit, and Moni Naor. Witnesses for Boolean matrix multiplication and for shortest paths. In *FOCS'92: Proceedings of the 33rd Annual IEEE Symposium on Foundations of Computer Science*, pages 417–426, 1992.

[2] Noga Alon and Moni Noar. Derandomization, witnesses for Boolean matrix multiplication and construction of perfect hash functions. *Algorithmica*, 16:434–449, 1996.

[3] Noga Alon, Raphael Yuster, and Uri Zwick. Color coding. *Journal of the ACM*, 42(4):844–856, 1995.

[4] Noga Alon, Raphael Yuster, and Uri Zwick. Finding and counting given length cycles. *Algorithmica*, 17:209–223, 1997.

[5] Ilya Baran, Erik D. Demaine, and Mihai Pătraşcu. Subquadratic algorithms for 3sum. *Algorithmica*, 50(4):584–596, 2008.

[6] E.R. Berlekamp, R.J. McEliece, and H.C.A. Van Tilborg. On the inherent intractability of certain coding problems.

IEEE Transactions on Information Theory, 24:384–386, 1978.

[7] Arnab Bhattacharyya, Victor Chen, Madhu Sudan, and Ning Xie. Testing linear-invariant non-linear properties. In *STACS'09*, pages 135–146, 2009.

[8] Arnab Bhattacharyya, Elena Grigorescu, and Asaf Shapira. A unified framework for testing linear-invariant properties. To appear in FOCS, 2010.

[9] Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. A duality between clause width and clause density for sat. In *CCC '06: Proceedings of the 21st Annual IEEE Conference on Computational Complexity*, pages 252–260, Washington, DC, USA, 2006. IEEE Computer Society.

[10] Marco Cesati. Compendium of parameterized problems, 2006.

[11] Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. Linear fpt reductions and computational lower bounds. In *STOC '04: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 212–221, New York, NY, USA, 2004. ACM.

[12] Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. On the computational hardness based on linear fpt-reductions. *J. Comb. Optim.*, 11(2):231–247, 2006.

[13] Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. Strong computational lower bounds via parameterized complexity. *J. Comput. Syst. Sci.*, 72(8):1346–1367, 2006.

[14] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19(90):297–301, 1965.

[15] Artur Czumaj and Andrzej Lingas. Finding a heaviest triangle is not harder than matrix multiplication. In *SODA'07: Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 986–994, 2007.

[16] Rodney G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness i: Basic results. *SIAM J. Comput.*, 24(4):873–921, 1995.

[17] Rodney G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness ii: On completeness for w[1]. *Theor. Comput. Sci.*, 141(1&2):109–131, 1995.

[18] Rodney G. Downey, Michael R. Fellows, Alexander Vardy, and Geoff Whittle. The parametrized complexity of some fundamental problems in coding theory. *SIAM J. Comput.*, 29(2):545–570, 1999.

[19] Ilya Dumer, Daniele Micciancio, and Madhu Sudan. Hardness of approximating the minimum distance of a linear code. *IEEE Transactions on Information Theory*, 49(1):22–37, 2003.

[20] Jeff Erickson. Bounds for linear satisfiability problems. *Chicago J. Theor. Comput. Sci.*, 1999, 1999.

[21] Jeff Erickson. New lower bounds for convex hull problems in odd dimensions. *SIAM J. Comput.*, 28(4):1198–1214, 1999.

[22] Jeff Erickson and Raimund Seidel. Better lower bounds on detecting affine and spherical degeneracies. *Discrete & Computational Geometry*, 13:41–57, 1995.

[23] Uriel Feige and Daniele Micciancio. The inapproximability of lattice and coding problems with preprocessing. *J. Comput. Syst. Sci.*, 69(1):45–67, 2004.

[24] F. Fomin, P. Golovach, D. Lokshtanov, and S. Saurabh. Algorithmic lower bounds for problems parameterized by clique-width. In *SODA*, 2010.

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

[26] Mira Gonen, Dana Ron, and Yuval Shavitt. Counting stars and other small subgraphs in sublinear time. In *SODA'10: Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2010. to appear.

[27] Ben Green. A Szemerédi-type regularity lemma in abelian groups, with applications. *Geom. Funct. Anal.*, 15(2):340–376, 2005.

[28] Venkatesan Guruswami and Alexander Vardy. Maximum-likelihood decoding of reed-solomon codes is np-hard. In *SODA*, pages 470–478, 2005.

[29] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.

[30] Tali Kaufman and Madhu Sudan. Algebraic property testing: The role of invariance. In *STOC'08: Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pages 403–412, 2008.

[31] James King. A survey of 3sum-hard problems, 2004.

[32] Daniel Král', Oriol Serra, and Lluís Vena. A removal lemma for systems of linear equations over finite fields, 2008.

[33] Daniel Král', Oriol Serra, and Lluís Vena. A combinatorial proof of the removal lemma for groups. *Journal of Combinatorial Theory*, 116(4):971–978, May 2009.

[34] D. Lokshtanov, Daniel Marx, and Saket Saurabh. Known algorithms on graphs of bounded treewidth are probably optimal, 2010.

[35] Dániel Marx. Can you beat treewidth? In *FOCS*, pages 169–179, 2007.

[36] Dániel Marx. On the optimality of planar and geometric approximation schemes. In *FOCS*, pages 338–348, 2007.

[37] David K. Maslen and Daniel N. Rockmore. Generalized FFTs - a survey of some recent results. In *Proceedings of the DIMACS Workshop on Groups and Computation*, pages 329–369, 1995.

[38] Joseph S. B. Mitchell and Joseph O'Rourke. Computational geometry column 42. *Int. J. Comput. Geometry Appl.*, 11(5):573–582, 2001.

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

[40] Mihai Patrascu. Towards polynomial lower bounds for dynamic problems. In *STOC*, pages 603–610, 2010.

[41] Mihai Pătraşcu and Ryan Williams. On the possibility of faster sat algorithms. In *SODA*, 2010.

[42] Oded Regev. Improved inapproximability of lattice and coding problems with preprocessing. *IEEE Transactions on Information Theory*, 50(9):2031–2037, 2004.

[43] Raimund Seidel. On the All-Pairs-Shortest-Path problem. In *STOC'92: Proceedings of the 24th Annual ACM Sym-*

posium on Theory of Computing, pages 745–749, 1992.

[44] Asaf Shapira. Green's conjecture and testing linear-invariant properties. In *STOC'09: Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, pages 159–166, 2009.

[45] Tamir Tassa and Jorge L. Villar. On proper secrets, (t,k)-bases and linear codes. *Des. Codes Cryptography*, 52(2):129–154, 2009.

[46] Alexander Vardy. Algorithmic complexity in coding theory and the minimum distance problem. In *STOC*, pages 92–109, 1997.

[47] Virginia Vassilevska and Ryan Williams. Finding a maximum weight triangle in $n^{3-\delta}$ time, with applications. In *STOC'06: Proceedings of the 38st Annual ACM Symposium on Theory of Computing*, pages 225–231, 2006.

[48] Virginia Vassilevska and Ryan Williams. Finding, minimizing, and counting weighted subgraphs. In *STOC'09: Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, pages 455–464, 2009.

[49] Ryan Williams. *Algorithms and Resource Requirements for Fundamental Problems*. PhD thesis, Carnegie Mellon University, 2007.