On the Hardness of Average-Case k-SUM

Zvika Brakerski
Weizmann Institute of Science, Rehovot, Israel

Noah Stephens-Davidowitz
Cornell University, Ithaca, NY, USA

Vinod Vaikuntanathan
Massachusetts Institute of Technology, Cambridge, MA, USA

Abstract

In this work, we show the first worst-case to average-case reduction for the classical k-SUM problem. A k-SUM instance is a collection of m integers, and the goal of the k-SUM problem is to find a subset of k integers that sums to 0. In the average-case version, the m elements are chosen uniformly at random from some interval $[-u, u]$.

We consider the *total* setting where m is sufficiently large (with respect to u and k), so that we are guaranteed (with high probability) that solutions must exist. In particular, $m = u^{\Omega(1/k)}$ suffices for totality. Much of the appeal of k-SUM, in particular connections to problems in computational geometry, extends to the total setting.

The best known algorithm in the average-case total setting is due to Wagner (following the approach of Blum-Kalai-Wasserman), and achieves a running time of $u^{\Theta(1/\log k)}$ when $m = u^{\Theta(1/\log k)}$. This beats the known (conditional) lower bounds for worst-case k-SUM, raising the natural question of whether it can be improved even further. However, in this work, we show a matching *average-case lower bound*, by showing a reduction from *worst-case lattice problems*, thus introducing a new family of techniques into the field of fine-grained complexity. In particular, we show that any algorithm solving average-case k-SUM on m elements in time $u^{o(1/\log k)}$ will give a super-polynomial improvement in the complexity of algorithms for lattice problems.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography

Keywords and phrases k-SUM, fine-grained complexity, average-case hardness

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2021.29

Category RANDOM

Related Version *Full Version*: `https://arxiv.org/abs/2010.08821`

Funding *Zvika Brakerski*: Supported by the Binational Science Foundation (Grant No. 2016726), and by the European Union Horizon 2020 Research and Innovation Program via ERC Project REACT (Grant 756482) and via Project PROMETHEUS (Grant 780701).
Noah Stephens-Davidowitz: Supported in part by NSF Grants CNS-1350619, CNS-1414119 and CNS-1718161, Microsoft Faculty Fellowship and an MIT/IBM grant.
Vinod Vaikuntanathan: Supported in part by NSF Grants CNS-1350619, CNS-1414119 and CNS-1718161, Microsoft Faculty Fellowship and an MIT/IBM grant.

1 Introduction

The k-SUM problem is a parameterized version of the classical subset sum problem. Given a collection of m integers $a_1, \ldots, a_m$, the k-SUM problem asks if there is some subset of cardinality k that sums to zero.[1] This problem (especially for $k = 3$, but more generally for

[1] This is the homogeneous version of k-SUM. One could also define the inhomogeneous version where the instance consists also of a target integer t, and the goal is to produce a subset of k elements that sums

arbitrary constant k) has been influential in computational geometry, where reductions from k-SUM have been used to show the conditional hardness of a large class of problems [16, 17]. More generally it has been used in computational complexity, where it has formed the basis for several fine-grained hardness results [33, 2, 38, 28]. We refer the reader to the extensive survey of Vassilevska-Williams [37] for an exposition of this line of work. The k-SUM problem has also been extensively studied in the cryptanalysis community (see, e.g., [36, 12, 10]).

We know two very different algorithms for k-SUM: a meet-in-the-middle algorithm that achieves run-time $O(m^{\lceil k/2 \rceil})$ [23], and dynamic programming or FFT-based algorithms that achieve run-time $\widetilde{O}(um)$ [11] where u is the largest absolute value of the integers a_i (A sequence of recent works [27, 14, 8, 25] improve the latter to $\widetilde{O}(u+m)$). Note that the latter algorithms outperform the former when $u \ll m^{\lceil k/2 \rceil}$, in what is sometimes called the *dense regime* of parameters, a point that we will come back to shortly.

In terms of hardness results for k-SUM, the work of Pătraşcu and Williams [34] shows that an algorithm that solves the problem in time $m^{o(k)}$ for all m will give us better algorithms for SAT, in particular refuting the exponential time hypothesis (ETH). The recent work of Abboud, Bringmann, Hermelin and Shabtay [1] shows that a $u^{1-\varepsilon}$-time algorithm (for any constant $\varepsilon > 0$) would refute the strong exponential-time hypothesis (SETH). So, we know that the two algorithms described above are essentially optimal, at least in the worst case.

The focus of this work is the natural average-case version of k-SUM where the problem instance $a_1, \ldots, a_m$ is chosen independently and uniformly at random from an interval $[-u, u]$. We call this the *average-case* k-SUM *problem*. In this setting, *deciding* whether a k-SUM solution exists is in many cases trivial. In particular, if $\binom{m}{k} \ll u$ then a union bound argument shows that the probability of a solution existing approaches 0. We refer to this as the *sparse* regime of the problem. In contrast, if $\binom{m}{k}$ is sufficiently larger than u, then a hashing argument guarantees the existence of many solutions, with high probability over the instance. (See Lemma 14.) As already mentioned above, we refer to this as the *dense* regime.

Notwithstanding this triviality, we notice that in the dense regime one could still consider the *search* problem of *finding* a k-SUM solution. The search problem seems to retain its hardness even in the dense setting and is the focus of our work. Since we consider the search version of the problem, we also refer to the dense regime as the *total* regime, as the associated search problem has a solution with high probability.

The average-case total problem is *not* quite as hard as the worst-case version (at least assuming SETH), since (slight variants of) Wagner's generalized birthday algorithm [36] and the Blum-Kalai-Wasserman algorithm [12] show how to solve this problem in time $u^{O(1/\log k)}$ (when $m = u^{\Omega(1/\log k)}$). This contrasts with the $u^{1-\varepsilon}$ lower bound of [1] in the worst case. (The BKW/Wagner algorithm was originally stated in a slightly different setting, so we restate it in Section 4.) This leaves the question of *how much easier* the average case is compared to the worst case. Given that the lower bounds from the worst-case setting are not a barrier here, it is a-priori unclear what is the best running time in this setting. Can we improve on [12, 36]?

1.1 Our Results

In this work we characterize the hardness of average-case k-SUM in the total regime by presenting a (conditional) lower bound that matches the $u^{O(1/\log k)}$ upper bound described above, up to the hidden constant in the exponent.

---

to t. In the worst-case world, the two versions are equivalent.

In more detail, our main result shows that average-case k-SUM is indeed hard to solve, under the assumption that *worst-case* lattice problems are hard to approximate. We thus introduce a new family of techniques into the study of the hardness of the k-SUM problem. Concretely, this lower bound shows that a $u^{o(1/\log k)}$-time algorithm for average-case k-SUM (in the dense regime) implies a $2^{o(n)}$-time $n^{1+\varepsilon}$-approximation algorithm for the shortest independent vectors problem (SIVP) over an n-dimensional lattice, a lattice problem for which the best known algorithms run in time $2^{\Omega(n)}$ [4, 3]. E.g., while Wagner's algorithm runs in time essentially linear in m when $m = u^{\Omega(1/\log k)}$, we show that such behavior for $m = u^{o(1/\log k)}$ would imply faster algorithms for SIVP. (One can think of our lower bound as ruling out linear-time algorithms for $m = u^{o(1/\log k)}$, or more generally ruling out $u^{o(1/\log k)}$-time algorithms for any m. Indeed, notice that the problem only gets easier as m becomes larger, so that lower bounds against linear-time algorithms for such large choices of m immediately implies lower bounds against $u^{o(1/\log k)}$-time algorithms for any m. We therefore switch freely between these two perspectives.)

It is widely believed that no faster algorithm can be found for SIVP. In particular, finding a $2^{o(n)}$-time algorithm for SIVP, would have major consequences in lattice-based cryptography both in theory and in practice [32, 6, 7].

We also note in the appendix that some of the connections between k-SUM and geometric problems from [16, 17] carry over to the dense setting as well. This shows an interesting (and not previously known, as far as we could find) connection between approximate short vectors in lattices, and computational geometry.

1.2 Our Techniques

The starting point of our reduction is the well known worst-case to average-case reductions in the lattice world, pioneered by Ajtai [5, 31, 19, 18]. These reductions show that the approximate shortest independent vectors (SIVP) problem, a standard problem in the lattice world, is at least as hard in the worst case as a certain problem called short integer solutions (SIS) on the average. The definition of lattices and the approximate shortest vector problem is not crucial for the current discussion, however we note again that the best algorithms on n-dimensional lattices that compute any poly(n)-approximation to SIVP run in time $2^{\Omega(n)}$. (We refer the curious reader to, e.g., [31, 35], Section 2.3, and the references therein for more background on lattices and lattice problems.)

In the (one-dimensional) *average-case* SIS problem with parameters m, Q and β, one is given random integers $a_1, \ldots, a_m \in \mathbb{Z}_Q$ and the goal is to find a *non-zero* integer linear combination $\boldsymbol{x} = (x_1, \ldots, x_m) \in \mathbb{Z}^m$ such that $\sum_{i\in[m]} a_i x_i = 0 \pmod{Q}$ and $\boldsymbol{x}$ is short, namely $||\boldsymbol{x}||_1 \leq \beta$. Thus, this is exactly the modular subset sum problem (i.e. subset sum over the group $\mathbb{Z}_Q$), except with weights larger than 1. The parameters of the problem live in the dense/total regime where such solutions are guaranteed to exist with high probability. The worst-case to average-case reductions state that an average-case SIS solver for a sufficently large Q, namely $Q = (\beta n)^{\Omega(n)}$, gives us an $\widetilde{O}(\sqrt{n \log m} \cdot \beta)$-approximate algorithm for SIVP. (We refer the reader to Theorem 10 for a more precise statement.)

Our main technical contribution is an average-case to average-case reduction from the SIS problem to the k-SUM problem. We show this by exhibiting a reduction from SIS to modular k-SUM (i.e. k-SUM over the group $\mathbb{Z}_Q$), and one from modular k-SUM to k-SUM. The latter is easy to see (in the dense regime): indeed, if you have a k-subset that sums to 0, it also sums to 0 $\pmod{Q}$ for any Q. Henceforth in this discussion, when we say k-SUM, we will mean modular k-SUM.

To reduce from SIS with parameters m, Q, β to k-SUM on m numbers over $\mathbb{Z}_Q$, we start with a simple, seemingly trivial, idea. SIS and k-SUM are so similar that perhaps one could simply run the k-SUM algorithm on the SIS instance. Unfortunately, this fails. For a k-SUM solution to exist, m has to be at least roughly $Q^{1/k} = n^{\Omega(n/k)}$. But, this could only possibly give us an approximate SIVP algorithm that runs in time $n^{\Omega(n/k)}$ (where we are most interested in constant k), since the reduction from SIVP has to at least write down the m samples. This is a meaningless outcome since, as we discussed before, there are algorithms for approximate SIVP that run in time $2^{O(n)}$.

Fortunately, ideas from the BKW algorithm [12] for subset sum (and the closely related algorithm from [36] for k-SUM) come to our rescue. We will start with SIS modulo $Q = q^L$ for some q and L that we will choose later. ([18] showed that worst-case to average-case reductions work for any sufficiently large Q, including $Q = q^L$.)

The BKW algorithm iteratively produces subsets that sum to 0 modulo q^i for $i = 1, \ldots, L$, finally producing SIS solutions modulo Q. To begin with, observe that for a k-subset-sum to exist modulo q, it suffices that $m \approx q^{1/k} \ll Q^{1/k}$, potentially getting us out of the conundrum from before. In particular, we will set $q \approx 2^{n \log k}$, $L \approx \log n / \log k$, therefore $Q = q^L \approx n^n$ as needed. We will also set $m \approx q^{\varepsilon / \log k} \approx 2^{\varepsilon n}$ for a large enough ε so that solutions exist (since $m^k \gg q$). Furthermore, a hypothetical k-SUM algorithm mod q that performs better than BKW/Wagner – that is, runs in time $q^{o(1/\log k)} = 2^{o(n)}$ – *is* potentially useful to us.

With this ray of optimism, let us assume that we can run the k-SUM algorithm many times to get several, m many, subsets S_j that sum to 0 modulo q. (We will return to, and remove, this unrealistic assumption soon.) That is,

$$b_j := \sum_{i \in S_j} a_i = 0 \pmod{q}$$

The BKW/Wagner approach would then be to use the $(b_1, \ldots, b_m)$ to generate $(c_1, \ldots, c_m)$ that are 0 (mod q^2), and so on. Note that c_i are a linear combination of $a_1, \ldots, a_m$ with weight k^2. At the end of the iterations, we will obtain at least one linear combination of $(a_1, \ldots, a_m)$ of weight $\beta = k^L$ that sums to 0 modulo $q^L = Q$, solving SIS. (We also need to make sure that this is a non-zero linear combination, which follows since the coefficients of all intermediate linear combinations are positive.)

This would finish the reduction, except that we need to remove our unrealistic assumption that we can use the k-SUM oracle to get many k-subsets of $(a_1, \ldots, a_m)$ that sum to 0. For one, the assumption is unrealistic because if we feed the k-SUM oracle with the same $(a_1, \ldots, a_m)$ (mod q) twice, we will likely get the same k-SUM solution. On the other hand, using a fresh random instance for every invocation of the k-SUM oracle will require m to be too large (essentially returning to the trivial idea above). A natural idea is to observe that each k-SUM solution touches a very small part of the instance. Therefore, one could hope to first receive a k-SUM $a_{i_1} + \ldots + a_{i_k}$ from the oracle, and in the next iteration, use as input $\{a_1, \ldots, a_m\} \setminus \{a_{i_1}, \ldots, a_{i_k}\}$, which is nearly as large as the original set. Unfortunately, continuing like this cannot work in general. The distributions of the successive instances that we feed to the oracle will no longer be uniform, and even worse, the oracle itself can choose which elements to remove from our set. A malicious oracle could therefore prevent us from obtaining many k-SUMs in this way, even if the oracle has high success probability on uniform input. (One can even imagine that the fastest algorithm for k-SUM would actually yield such a malicious oracle. E.g., if an algorithm has a "preference" for some types of k-SUMs over others, then running the algorithm repeatedly in this way could eventually deplete the input of such k-SUMs, causing the algorithm to fail.)

Instead, our key idea is rather simple, namely to resort to (re)randomization. Given an instance $(a_1, \ldots, a_m) \in \mathbb{Z}_q^m$, we compute many random subset sums to generate $(a'_1, \ldots, a'_m) \in \mathbb{Z}_q^m$. That is, we choose k-subsets $T'_i \subseteq [m]$ and let

$$a'_i = \sum_{j \in T'_i} a_j \pmod{q}$$

Since $q \gg m^{1/k}$, the leftover hash lemma [24] tells us that the a'_i are (statistically close to) uniformly random mod q. Furthermore, a k-subset sum of $(a'_1, \ldots, a'_m)$ will give us a k^2-subset sum of $(a_1, \ldots, a_m)$ that sums to 0 (mod q). To obtain a new subset sum of $(a_1, \ldots, a_m)$, simply run this process again choosing fresh subsets T''_i to generate $(a''_1, \ldots, a''_m)$; and so on. Eventually, this will give us a $\beta = k^{2L}$ weight solution to SIS, which is a quadratic factor worse than before, but good enough for us. (We are glossing over an important technical detail here, which is how we ensure that the resulting subset sums yield uniformly random independent elements in $q\mathbb{Z}/q^2\mathbb{Z}$.)

To finish the analysis of the reduction, observe that it calls the k-SUM oracle $\approx mL$ times. Assuming the oracle runs in time $q^{o(1/\log k)}$, this gives us a $2^{o(n)}$-time algorithm for approximate SIVP. The approximation factor is $\widetilde{O}(\sqrt{n \log m} \cdot \beta) \approx n^3$. (In the sequel, we achieve $n^{1+o(1)}$ by a careful choice of parameters.)

Interestingly, our reduction re-imagines the BKW/Wagner *algorithm as a reduction* from the SIS problem to k-SUM, where the original algorithm is achieved (in retrospect) by plugging in the trivial algorithm for 2-SUM. Of course, the algorithm is much simpler than the reduction (in particular, there is no need for re-randomization) since we don't need to account for "malicious" k-SUM solvers. Our main technical contribution can therefore be viewed as making the ideas from the BKW/Wagner *algorithm* (ideas which are now ubiquitous in the study of algorithms for subset sum and lattice problems) work as a *reduction* – i.e., with an arbitrary average-case k-SUM oracle.

1.3 Open Problems and Future Directions

Our work introduces the powerful toolkit of lattice problems into the field of average-case fine-grained complexity, and raises several natural directions for further research.

First is the question of whether a result analogous to what we show holds in the sparse/-planted regime as well. A possible theorem here would rule out an $m^{o(k)}$-time algorithm for k-SUM, assuming the hardness of lattice problems. To the best of our knowledge, in the sparse/planted regime it is not known whether the average-case problem is easier than the worst-case problem as in the dense regime.

Second is the question of whether we can obtain average-case hardness of k-SUM for concrete small constants k, perhaps even $k = 3$. Our hardness result is asymptotic in k.

Third is the question of whether we can show the average-case hardness of natural distributions over *combinatorial and computational-geometric* problems, given their connection to k-SUM. In this vein, we show a simple reduction to (perhaps not the most natural distribution on) the (Q, m, d)-plane problem in Appendix A, but we believe much more can be said. More generally, now that we have shown average-case hardness of k-SUM, it is natural to try to reduce average-case k-SUM to other natural average-case problems.

1.4 Other Related Works

There are now quite a few works that study average-case fine-grained hardness of problems in P. We mention a few. First, Ball, Rosen, Sabin, and Vasudevan [9] showed a reduction from SAT to an (average-case) variant of the orthogonal vectors problem. They demonstrated that sub-quadratic algorithms for their problem would refute SETH.

There is also a sequence of works on the average-case hardness of counting k-cliques. The work of Goldreich and Rothblum [20, 21] shows worst-case to average-case *self*-reductions for the problem of counting k-cliques (and other problems in P). Boix-Adserà, Brennan, and Bresler proved the same result for $G_{n,p}$ [13], and Hirahara and Shimizu recently showed that it is even hard to count the number of k-cliques with even a small probability of success [22]. In contrast, our reductions go from the worst-case of one problem (SIVP) to the average-case of another (k-SUM). We find it a fascinating problem to show a worst-case to average-case *self*-reduction for k-SUM.

Dalirrooyfard, Lincoln, and Vassilevska Williams [15] recently proved fine-grained average-case hardness for many different problems in P under various complexity-theoretic assumptions. In particular, they show fine-grained average-case hardness of *counting* the number of solutions of a non-standard *factored* variant of k-SUM under well-studied fine-grained hardness assumptions. In contrast, we show fine-grained average-case hardness of the standard *search* k-SUM problem under an assumption that is well-studied in the lattice community but perhaps not previously considered in the fine-grained complexity world. So, our conclusion is more natural – both because it works for a search problem rather than a counting problem and because it works with the standard notion of k-SUM rather than a factored variant – but we rely on a less-standard assumption.

For the lattice expert, we remark that if one unwraps our reduction from SIVP to SIS and then to k-SUM, we obtain a structure that is superficially similar to [30]. However, in their setting, they do not need to reuse samples and therefore do not need the re-randomization technique, which is the key new idea in this work.

More generally, there are many works that use BKW/Wagner-style techniques together with a specific solver for k-SUM or subset sum to solve various lattice problems. (See, for example, [29, 30, 26].) In contrast, we show a generic *reduction* from SIVP to average-case k-SUM that can be instantiated with *any* average-case k-SUM oracle.

1.5 Organization of the Paper

Section 3 describes the modular variant of k-SUM as well as the standard k-SUM (over the integers), shows their totality on average, and reductions between them. For completeness, we describe the BKW/Wagner algorithm in Section 4. We remark that while the standard descriptions of the algorithm refer to finite groups, we need one additional trick (namely, Lemma 15) to obtain the algorithm over the integers. Finally, our main result, the worst-case to average-case reduction is described in Section 5. The connection to computational geometry is provided in Appendix A.

2 Preliminaries

We write log for the logarithm base two and ln for the natural logarithm. We write $\binom{m}{k} := \frac{m!}{(m-k)!k!}$ for the binomial coefficient.

2.1 Probability

We make little to no distinction between random variables and their associated distributions.

For two random variables X, Y over some set S, we write $\Delta(X,Y) := \sum_{z\in S} |\Pr[X = z] - \Pr[Y = z]|$ for the statistical distance between X and Y. For a finite set S, we write U_S for the uniform distribution over S.

Recall that a set of functions $\mathcal{H} \subseteq \{h : X \to Y\}$ is a *universal family of hash functions* from X to Y if for any distinct $x, x' \in X$

$$\Pr_{h\sim\mathcal{H}}[h(x) = h(x')] \leq 1/|Y| \ .$$

▶ **Lemma 1** (Leftover hash lemma, [24]). *If $\mathcal{H}$ is a universal family of hash functions from X to Y, then*

$$\Pr[\Delta(h(U_X), U_Y) \geq \beta] \leq \beta \ ,$$

where the probability is over a random choice of $h \sim \mathcal{H}$ and $\beta := (|Y|/|X|)^{1/4}$.

▶ **Lemma 2.** *For any positive integers Q, m, let $\mathcal{H}$ be the family of hash functions from $\{0,1\}^m$ to $\mathbb{Z}_Q$ given by $h_{\boldsymbol{a}}(\boldsymbol{x}) = \langle \boldsymbol{a}, \boldsymbol{x}\rangle \bmod Q$ for all $\boldsymbol{a} \in \mathbb{Z}_Q^m$. Then, $\mathcal{H}$ is a universal family of hash functions.*

Proof. Let $\boldsymbol{x}, \boldsymbol{y} \in \{0,1\}^m$ be distinct vectors, and suppose without loss of generality that $x_1 = 1$ and $y_1 = 0$. We write $\boldsymbol{a}' \in \mathbb{Z}_Q^{m-1}$ for the vector obtained by removing the first coordinate from $\boldsymbol{a}$ and a_1 for the first coordinate itself. Similarly, we write $\boldsymbol{x}', \boldsymbol{y}' \in \{0,1\}^{m-1}$ for the vectors $\boldsymbol{x}, \boldsymbol{y}$ with their first coordinate removed. Then,

$$\begin{aligned}\Pr[\langle \boldsymbol{a}, \boldsymbol{x}\rangle = \langle \boldsymbol{a}, \boldsymbol{y}\rangle \bmod Q] &= \Pr[\langle \boldsymbol{a}', \boldsymbol{x}'\rangle + a_1 = \langle \boldsymbol{a}', \boldsymbol{y}'\rangle \bmod Q] \\ &= \Pr[a_1 = \langle \boldsymbol{a}', \boldsymbol{y}' - \boldsymbol{x}'\rangle \bmod Q] = 1/Q \ ,\end{aligned}$$

where the probability is over the random choice of $\boldsymbol{a} \in \mathbb{Z}_Q^m$. The last equality follows from the fact that $a_1 \in \mathbb{Z}_Q$ is uniformly random and independent of $\boldsymbol{a}'$. ◀

▶ **Corollary 3.** *For any positive integers Q, m and any subset $X \subseteq \{0,1\}^m$,*

$$\Pr_{\boldsymbol{a}\sim\mathbb{Z}_Q^m}[\Delta(\langle \boldsymbol{a}, U_X\rangle \bmod Q, U_{\mathbb{Z}_Q}) \geq \beta] \leq \beta \ ,$$

where $\beta := (Q/|X|)^{1/4}$.

▶ **Corollary 4.** *Let $\boldsymbol{a} := (a_1, \ldots, a_M) \in \mathbb{Z}_Q^M$ be sampled uniformly at random, and let $S_1, \ldots, S_{M'} \subset [M]$ be sampled independently and uniformly at random with $|S_i| = t$. Let $c_i := \sum_{j\in S_i} a_j \bmod Q$. Then, $(\boldsymbol{a}, \boldsymbol{c}) := (a_1, \ldots, a_M, c_1, \ldots, c_{M'})$ is within statistical distance δ of a uniformly random element in $\mathbb{Z}_Q^{M+M'}$, where*

$$\delta := (M'+1) \cdot Q^{1/4} \cdot \binom{M}{t}^{-1/4} \leq (M'+1) \cdot \left(\frac{Qt^t}{M^t}\right)^{1/4} \ .$$

Proof. Let $X_t := \{\boldsymbol{x} \in \{0,1\}^M \ : \ \|\boldsymbol{x}\|_1 = t\}$, and notice that $|X_t| = \binom{M}{t}$. Call $\boldsymbol{a}$ good if $\Delta(\langle \boldsymbol{a}, U_{X_t}\rangle \bmod Q, U_{\mathbb{Z}_Q}) \leq \beta := Q^{1/4}/|X_t|^{1/4}$. From Corollary 3, we see that $\boldsymbol{A}$ is good except with probability at most β.

Finally, notice that the c_i are distributed exactly as independent samples from $\langle \boldsymbol{a}, U_{X_t}\rangle$. Therefore, if $\boldsymbol{a}$ is good, each of the c_i is within statistical distance β of an independent uniform sample. The result then follows from the union bound. ◀

2.2 Hitting probabilities

▶ **Definition 5.** *For $\boldsymbol{a} := (a_1, \ldots, a_M) \in \mathbb{Z}_Q^M$, $\boldsymbol{c} := (c_1, \ldots, c_{M'}) \in \mathbb{Z}_Q^{M'}$, $I \subset [M]$, $J \subset [M']$, and a positive integer t, the* t-hitting probability *of $\boldsymbol{a}$, $\boldsymbol{c}$, I, and J is defined as follows. For each $j \in J$, sample a uniformly random $S_j \in \binom{[M]}{t}$ with $\sum_{i \in S_j} a_i = c_j$. (If no such S_j exists, then we define the hitting probability to be 1.) The hitting probability is then*

$$p_{\boldsymbol{a},\boldsymbol{c},I,J,t} := \Pr[\exists\, j, j' \in J \text{ such that } S_j \cap I \neq \emptyset \text{ or } S_j \cap S_{j'} \neq \emptyset] \,.$$

▶ **Lemma 6.** *For any positive integers Q, M, t and $0 < \varepsilon < 1$,*

$$\Pr_{\boldsymbol{a} \sim \mathbb{Z}_Q^M, c \sim \mathbb{Z}_Q} \left[p_{\boldsymbol{a},c,t} \geq \frac{1+\varepsilon}{1-\varepsilon} \cdot \frac{t}{M} \right] \leq \frac{4Q^{1/4}}{\varepsilon \cdot \binom{M-1}{t-1}^{1/4}} \,.$$

where

$$p_{\boldsymbol{a},c,t} := p_{\boldsymbol{a},c,\{1\},\{1\},t} \,.$$

Proof. We have

$$p_{\boldsymbol{a},c,t} = \Pr_{\boldsymbol{x} \sim X_t}[x_1 = 1 \mid \langle \boldsymbol{a}, \boldsymbol{x} \rangle = c \bmod Q] \,,$$

where $X_t := \{\boldsymbol{x} \in \{0,1\}^M \ : \ \|\boldsymbol{x}\|_1 = t\}$. Therefore,

$$\begin{aligned} p_{\boldsymbol{a},c,t} &= \Pr_{\boldsymbol{x} \sim X_t}[x_1 = 1] \cdot \Pr_{\boldsymbol{x} \sim X_t}[\langle \boldsymbol{a}, \boldsymbol{x} \rangle = c \bmod Q \mid x_1 = 1] / \Pr_{\boldsymbol{x} \sim X_t}[\langle \boldsymbol{a}, \boldsymbol{x} \rangle = c \bmod Q] \\ &= \frac{t}{M} \cdot \Pr_{\boldsymbol{x}' \sim X'_{t-1}}[\langle \boldsymbol{a}_{-1}, \boldsymbol{x}' \rangle = c - a_1 \bmod Q] / \Pr_{\boldsymbol{x} \sim X_t}[\langle \boldsymbol{a}, \boldsymbol{x} \rangle = c \bmod Q] \,, \end{aligned}$$

where $\boldsymbol{a}_{-1}$ is $\boldsymbol{a}$ with its first coordinate removed and $X'_{t-1} := \{\boldsymbol{x} \in \{0,1\}^{M-1} \ : \ \|\boldsymbol{x}\|_1 = t-1\}$.

So, let

$$p_{\boldsymbol{a},c} := \Pr_{\boldsymbol{x} \in X_t}[\langle \boldsymbol{a}, \boldsymbol{x} \rangle = c \bmod Q] \,,$$

and

$$p'_{\boldsymbol{a},c} := \Pr_{\boldsymbol{x}' \in X'_{t-1}}[\langle \boldsymbol{a}_{-1}, \boldsymbol{x}' \rangle = c - a_1 \bmod Q] \,.$$

As in the proof of Corollary 4, we see that

$$\sum_{c \in \mathbb{Z}_Q} |p_{\boldsymbol{a},c} - 1/Q| = \Delta(\langle \boldsymbol{a}, U_{X_t} \rangle \bmod Q, U_{\mathbb{Z}_Q}) \leq Q^{1/4} / \binom{M}{t}^{1/4} \tag{1}$$

except with probability at most $Q^{1/4}/\binom{M}{t}^{1/4}$ over $\boldsymbol{a}$. Similarly,

$$\sum_{c \in \mathbb{Z}_Q} |p'_{\boldsymbol{a},c} - 1/Q| = \Delta(\langle \boldsymbol{a}_{-1}, U_{X'_{t-1}} \rangle \bmod Q, U_{\mathbb{Z}_Q}) \leq Q^{1/4} / \binom{M-1}{t-1}^{1/4} \tag{2}$$

except with probability at most $Q^{1/4}/\binom{M-1}{t-1}^{1/4}$ over $\boldsymbol{a}$.

So, suppose that $\boldsymbol{a}$ satisfies Eq. (1) and Eq. (2). Then, by Markov's inequality,

$$\Pr_{c \in \mathbb{Z}_Q}[p_{\boldsymbol{a},c} \geq (1-\varepsilon)/Q] \leq \frac{Q^{1/4}}{\varepsilon \cdot \binom{M}{t}^{1/4}}$$

for any $0 < \varepsilon < 1$, and similarly,

$$\Pr_{c \in \mathbb{Z}_Q}[p'_{\boldsymbol{a},c} \leq (1+r)/Q] \leq \frac{Q^{1/4}}{\varepsilon \cdot \binom{M-1}{t-1}^{1/4}} .$$

Therefore, for such $\boldsymbol{a}$,

$$\Pr[p_{\boldsymbol{a},c,t} \geq (1+\varepsilon)t/((1-\varepsilon)M)] \leq \frac{2Q^{1/4}}{\varepsilon \cdot \binom{M-1}{t-1}^{1/4}} .$$

The result then follows by union bound. ◀

By repeated applications of union bound, we derive the following corollary.

▶ **Corollary 7.** *For any positive integers Q, M, M', t, v, v' and $0 < \varepsilon < 1$, let $\boldsymbol{a} \sim \mathbb{Z}_Q^M$ and $\boldsymbol{c} \sim \mathbb{Z}_Q^{M'}$ be sampled uniformly at random. Then,*

$$p_{\boldsymbol{a},\boldsymbol{c},I,J,t} \leq (v + tv') \cdot v' \cdot \frac{1+\varepsilon}{1-\varepsilon} \cdot \frac{t}{M}$$

for all $I \in \binom{[M]}{\leq v}$, $J \in \binom{[M']}{\leq v'}$ except with probability at most

$$4MM' \frac{Q^{1/4}}{\varepsilon \cdot \binom{M-1}{t-1}^{1/4}} .$$

Proof. Let $\eta := \max_{i,j} p_{\boldsymbol{a},\boldsymbol{c},\{i\},\{j\},t}$. By union bound, for any set I, we have

$$p_{\boldsymbol{a},\boldsymbol{c},I,\{j\},t} \leq \sum_{i \in I} p_{\boldsymbol{a},\boldsymbol{c},\{i\},\{j\},t} \leq |I| \cdot \eta .$$

Fix some set J. Let S_j be as in the definition of the hitting probability, and let $I_{-j} := I \cup \bigcup_{j' \in J \setminus \{j\}} S_{j'}$. Notice that $|I_{-j}| \leq |I| + t|J|$. Then,

$$p_{\boldsymbol{a},\boldsymbol{c},I,J,t} \leq \sum_{j \in J} p_{\boldsymbol{a},\boldsymbol{c},I_{-j},\{j\},t} \leq (|I| + t|J|) \cdot |J| \cdot \eta .$$

Finally, by union bound and Lemma 6, we have

$$\eta \leq \frac{1+\varepsilon}{1-\varepsilon} \cdot \frac{r}{M}$$

except with probability at most

$$4MM' \frac{Q^{1/4}}{\varepsilon \cdot \binom{M-1}{t-1}^{1/4}} .$$

The result follows. ◀

2.3 Lattices and Lattice Problems

▶ **Definition 8** (Shortest Independent Vectors Problem)**.** *For an approximation factor $\gamma := \gamma(n) \geq 1$, γ-SIVP is the search problem defined as follows. Given a lattice $\mathcal{L} \subset \mathbb{R}^n$, output n linearly independent lattice vectors which all have length at most $\gamma(n)$ times the minimum possible, $\lambda_n(\mathcal{L})$.*

▶ **Definition 9** (Short Integer Solutions). *For integers m, Q, β, the (average-case) short integer solutions problem* $\mathsf{SIS}(m, Q, \beta)$ *is defined by m integers $a_1, \ldots, a_m$ drawn uniformly at random and independently from $\mathbb{Z}_Q$, and the goal is to come up with a* non-zero *vector $\mathbf{x} = (x_1, \ldots, x_m)$ where*

$$\sum_{i \in [m]} x_i a_i = 0 \pmod{Q} \quad \text{and} \quad ||\mathbf{x}||_1 := \sum_{i=1}^{m} |x_i| \leq \alpha$$

Following the seminal work of Ajtai [5], there have been several works that show how to solve the *worst-case* γ-SIVP problem given an algorithm for the *average-case* SIS problem. We will use the most recent one due to Gama et al. [18] (specialized to the case of cyclic groups for simplicity).

▶ **Theorem 10** (Worst-Case to Average-Case Reduction for SIS [31, 18]). *Let $n, Q, \beta \in \mathbb{N}$ where $Q = (\beta n)^{\Omega(n)}$. If there is an algorithm for the* average-case *SIS problem* $\mathsf{SIS}(m, Q, \beta)$ *over $\mathbb{Z}_Q$ that runs in time T, then there is an $(m+T) \cdot \mathsf{poly}(n)$-time algorithm for* worst-case *$\widetilde{O}(\sqrt{n \log m} \cdot \beta)$-SIVP on any n-dimensional lattice L.*

3 Variants of Average-case k-SUM: Totality and Reductions

We define two variants of average-case k-SUM, one over the integers (which is the standard version of k-SUM) and one over the finite group $\mathbb{Z}_Q$ of integers modulo Q. We show that the hardness of the two problems is tied together, which will allow us to use the modular version for our results down the line.

▶ **Definition 11** (Average-case k-SUM). *For positive integers $m, k \geq 2$ and $u \geq 1$, the* average-case k-SUM(u, m) *problem is the search problem defined as follows. The input is $a_1, \ldots, a_m \in [-u, u]$ chosen uniformly and independently at random, and the goal is to find k distinct indices $i_1, \ldots, i_k$ such that $a_{i_1}, \ldots, a_{i_k}$ with $a_{i_1} + \cdots + a_{i_k} = 0$.*

We define the modular version of the problem where the instance consists of numbers chosen at random from the finite additive group $\mathbb{Z}_Q$ of numbers modulo Q. This will appear as an intermediate problem in our algorithm in Section 4 and our worst-case to average-case reduction in Section 5.

▶ **Definition 12** (Average-case Modular k-SUM). *For integers $m, k \geq 2$ and integer modulus $Q \geq 2$, the average-case k-SUM$(\mathbb{Z}_Q, m)$ problem is the search problem defined as follows. The input is $a_1, \ldots, a_m \sim \mathbb{Z}_Q$ chosen uniformly and independently at random, and the goal is to find k distinct indices $i_1, \ldots, i_k$ such that $a_{i_1}, \ldots, a_{i_k}$ with $a_{i_1} + \cdots + a_{i_k} = 0 \pmod{Q}$.*

We highlight the distinction in our notation for the two problems. The former (non-modular version) is denoted k-SUM(u, m) (the first parameter is the bound u on the absolute value of the elements), whereas the latter is denoted k-SUM$(\mathbb{Z}_Q, m)$ (the first parameter indicates the group on which the problem is defined). The second parameter always refers to the number of elements in the instance.

We now show that the modular problem is total when $\binom{m}{k} \gtrsim Q$ and is unlikely to have a solution when $\binom{m}{k} \lesssim Q$.

▶ **Lemma 13.** *If $a_1, \ldots, a_m \sim \mathbb{Z}_Q$ are sampled uniformly at random, and E_k is the event that there exist distinct indices $i_1, \ldots, i_k$ with $a_{i_1} + \cdots + a_{i_k} = 0 \pmod{Q}$, then*

$$1 - Q/\binom{m}{k} \leq \Pr[E_k] \leq \binom{m}{k}/Q \,.$$

Proof. Notice that for fixed indices $i_1, \ldots, i_k$, the probability that $a_{i_1} + \cdots + a_{i_k} = 0$ is exactly $1/Q$. The upper bound then follows from a union bound over all $\binom{m}{k}$ k-tuples of indices. Furthermore, notice that $i_1, \ldots, i_k$ and $j_1, \ldots, j_k$, the event that $a_{i_1} + \cdots + a_{i_k} = 0 \pmod{Q}$ is independent of the event that $a_{j_1} + \cdots + a_{j_k} = 0 \pmod{Q}$ as long as $\{i_1, \ldots, i_k\} \neq \{j_1, \ldots, j_k\}$. The lower bound then follows from Chebyshev's inequality. ◀

▶ **Lemma 14.** *If $a_1, \ldots, a_m \sim [-u, u]$ are sampled uniformly at random, and E_k is the event that there exist distinct indices $i_1, \ldots, i_k$ with $a_{i_1} + \cdots + a_{i_k} = 0$, then*

$$1 - e^{-\alpha} \leq \Pr[E_k] \leq \binom{m}{k}/(2u+1) \ ,$$

where

$$\alpha := \frac{1}{4k+2} \cdot \left\lfloor \frac{m}{k(20u+10)^{1/k}} \right\rfloor \approx m/(k^2 u^{1/k}) \ .$$

Proof. The upper bound follows immediately from the upper bound in Lemma 13 together with the observation that elements that sum to zero over the integers must sum to zero modulo $Q := 2u + 1$ as well.

Let $m' := k(10Q)^{1/k}$. Let E'_k be the event that there exist distinct indices $i_1, \ldots, i_k \leq m'$ with $a_{i_1} + \cdots + a_{i_k} = 0$. Notice that

$$\Pr[E_k] \geq 1 - (1 - \Pr[E'_k])^{\lfloor m/m' \rfloor} \geq 1 - \exp(-\lfloor m/m' \rfloor \Pr[E'_k]) \ .$$

So, it suffices to show that

$$\Pr[E'_k] \geq \frac{1 - Q/\binom{m'}{k}}{2k+1} \geq \frac{1}{4k+2} \ .$$

By Lemma 13, we know that with probability at least $1 - Q/\binom{m'}{k}$, there exists a k-SUM that sums to zero modulo Q in the first m' elements. I.e., $a_{i_1} + \cdots + a_{i_k} = \ell Q$ for some $\ell \in \{-k, -k+1, \ldots, k-1, k\}$ and $i_1, \ldots, i_k \leq m'$. We wish to argue that $\ell = 0$ is at least as likely as $\ell = i$ for any i.

Let $p(k', s) := \Pr[a_1 + \cdots + a_{k'} = s]$ for integers k', s. Notice that for $s \geq 0$, we have

$$\begin{aligned} p(k'+1, s) - p(k'+1, s+1) &= \big(p(k', -(s+u)) - p(k', s+u+1)\big)/(2u+1) \\ &= \big(p(k', s+u) - p(k', s+u+1)\big)/(2u+1) \ . \end{aligned}$$

It then follows from a simple induction argument that $p(k, s+1) \leq p(k, s)$. In particular, $p(k, \ell Q) \leq p(k, 0)$ for any ℓ. Therefore, letting $E'_{k,Q}$ be the event that the first m' elements contain a k-SUM modulo Q, we have

$$\begin{aligned} \Pr[E'_k] &\geq \Pr[E'_{k,Q}] \cdot \Pr[a_{i_1} + \cdots + a_{i_k} = 0 \mid a_{i_1} + \cdots + a_{i_k} = 0 \pmod{Q}] \\ &\geq \frac{\Pr[E'_{k,Q}]}{2k+1} \ . \end{aligned}$$

Finally, by Lemma 13, we have

$$\Pr[E'_{k,Q}] \geq 1 - Q/\binom{m'}{k} \geq 1/2 \ ,$$

as needed. ◀

3.1 From k-SUM to Modular k-SUM and Back

We first show that an algorithm for the modular k-SUM problem gives us an algorithm for the k-SUM problem. A consequence of this is that when we describe the algorithm for k-SUM in Section 4, we will focus on the modular variant.

▶ **Lemma 15.** *Let u be a positive integer and let $Q = 2u + 1$. If there is an algorithm for the k-SUM$(\mathbb{Z}_Q, m)$ that runs in time T and succeeds with probability p, then there is an algorithm for $2k$-SUM$(u, 2m)$ that runs in time $O(T)$ and succeeds with probability at least p^2/k.*

Proof. Let $\mathcal{A}$ be the purported algorithm for k-SUM$(\mathbb{Z}_Q, m)$. The algorithm for $2k$-SUM$(u, 2m)$ receives $2m$ integers $a_1, \ldots, a_{2m}$ in the range $[-u, u]$ and works as follows. We use the natural embedding to associate elements in $\mathbb{Z}_Q$ with elements in $[-u, u]$, so we may think of $a_1, \ldots, a_{2m}$ also as elements in $\mathbb{Z}_Q$ (simply by considering their coset modulo Q).

- Run $\mathcal{A}$ on $a_1, \ldots, a_m$ to obtain a k-subset S_1. If $\mathcal{A}$ does not succeed, then fail.
- Run $\mathcal{A}$ on $-a_{m+1}, \ldots, -a_{2m}$ to obtain a k-subset S_2. If $\mathcal{A}$ does not succeed, then fail.
- If $\sum_{i \in S_1} a_i = -\sum_{i \in S_2} a_i$, output $S_1 \cup S_2$ as the $2k$-subset. Fail otherwise.

It is clear that the run-time is $O(T)$ and that if the algorithm does not fail then it indeed outputs a valid $2k$-sum. It suffices to bound the probability that the algorithm succeeds.

Since the first two steps run $\mathcal{A}$ on independent and identically distributed input, we can deduce that the probability that both succeed is p^2, and in the case that both succeed, their output satisfies

$$\sum_{i \in S_1} a_i = \alpha_1 Q \quad \text{and} \quad \sum_{i \in S_1} a_i = \alpha_2 Q$$

for some integers $\alpha_1, \alpha_2 \in (-k/2, k/2)$, which are independent and identically distributed random variables. The probability that $\alpha_1 = \alpha_2$ is therefore at least $1/k$, since the collision probability of a random variable is bounded by the inverse of its support size. If this happens then, $\sum_{i \in S_1} a_i = \sum_{i \in S_2} a_i$ and the algorithm succeeds. Thus, we conclude that our algorithm succeeds with probability at least p^2/k. ◀

Finally, we show a proof in the other direction. Namely, that an algorithm for the k-SUM problem gives us an algorithm for the modular k-SUM problem. We will use this when we describe the worst-case to average-case reduction in Section 5.

▶ **Lemma 16.** *For $m \geq k \cdot u^{2/k}$, if there is an algorithm for k-SUM(u, m) that runs in time T and succeeds with probability p, then there is an algorithm for k-SUM$(\mathbb{Z}_{2u+1}, m)$ that runs in time T and succeeds with probability p.*

Proof. Let $\mathcal{A}$ be the purported algorithm for k-SUM(u, m). The algorithm for k-SUM$(\mathbb{Z}_{2u+1}, m)$ receives m integers $a_1, \ldots, a_{2m} \in \mathbb{Z}_{2u+1}$ and works as follows.

As before, identify $\mathbb{Z}_{2u+1}$ with the interval $[-u, u]$ and run $\mathcal{A}$ on $a_1, \ldots, a_m$. We can then simply output the resulting k-subset S. In particular, if $\sum_{i \in S} a_i = 0$, then we of course have $\sum_{i \in S} a_i = 0 \bmod 2u + 1$.

Clearly, the success probability of the resulting algorithm is at least p, since the input to $\mathcal{A}$ is distributed uniformly. ◀

4 The $u^{O(1/\log k)}$-time Algorithm for Average-case k-SUM

In this section, we describe a variant of the Blum-Kalai-Wasserman algorithm [12] for the average-case k-SUM problem that runs in time $u^{O(1/\log k)}$.

▶ **Theorem 17.** *There is a $\widetilde{O}(2^\ell q^2)$-time algorithm that solves average-case 2^ℓ-SUM$(\mathbb{Z}_{q^\ell}, m)$ for $m = \widetilde{\Theta}(2^\ell q^2)$.*

Proof. On input $a_1, \ldots, a_m \in \mathbb{Z}_{q^\ell}$ with $m := 1000\ell^2 q^2 2^\ell \log q = \widetilde{\Theta}(2^\ell q^2)$, the algorithm behaves as follows. Let $L_1 := (a_1, \ldots, a_m)$. For $i = 1, \ldots, \ell$, the algorithm groups the elements in L_i according to their value modulo q^i. It then greedily groups them into m_{i+1} disjoint points (a, b) with $a + b = 0 \bmod q^i$. It sets L_{i+1} to be the list of sums of these pairs (and records the indices of the 2^i input elements that sum to $a + b$). If at any point the algorithm fails to find such pairs, it simply fails; otherwise, the algorithm outputs the elements $a_{i_1}, \cdots, a_{i_{2^\ell}}$ satisfying $\sum a_{i_j} = 0 \bmod q^\ell$ found in the last step.

The running time of the algorithm is clearly $\mathrm{poly}(\ell, \log q, \log m)m$ as claimed. To prove correctness, we need to show that at each step the algorithm is likely to succeed in populating the list L_{i+1} with at least $m_i := (\ell^2 - i^2)/\ell^2 \cdot m/2^{i-1}$ elements, since clearly the algorithm outputs a valid 2^ℓ-SUM in this case.

Suppose that the algorithm succeeds up to the point where it populates L_i. Let $L_i = (b_1, \ldots, b_{m_i})$, and $b'_i := (b_i/q^{i-1}) \bmod q$, where the division by q^{i-1} is possible because $b_i = 0 \bmod q^{i-1}$ by assumption. Notice that the b'_i are independent and uniformly random. For $j \in \mathbb{Z}_q$, let $c_j := |\{i \ : \ b'_i = j \bmod q\}|$. Notice that the algorithm successfully populates L_{i+1} if and only if

$$\sum_{j \in \mathbb{Z}_q} \min\{c_j, c_{-j}\}/2 \geq m_{i+1} \ .$$

By the Chernoff-Hoeffding bound, we have that

$$\Pr\left[c_j < m_i/q - 10\sqrt{m_i \log m_i}\right] \leq 1/m_i^2$$

It follows that

$$\sum_j \min\{c_j, c_{-j}\}/2 \geq q \min\{c_j\}/2 \geq m_i/2 - 5q\sqrt{m_i \log m_i} \geq m_{i+1}$$

except with probability at most $1/m_i$. By union bound, we see that the algorithm succeeds in populating every list except with probability at most $\sum 1/m_i \ll 1/10$, as needed. ◀

Combining this with Lemma 15 (the reduction from k-SUM to modular k-SUM), we obtain the following corollary.

▶ **Corollary 18.** *For $u = (q^\ell - 1)/2$ for odd q and $k = 2^{\ell+1}$, there is a $u^{O(1/\log k)}$-time algorithm for k-SUM(u, m) for $m = u^{\Theta(1/\log k)}$.*

5 From Worst-case Lattice Problems to Average-case k-SUM

In this section, we describe our main result, namely a worst-case to average-case reduction for k-SUM. We state the theorem below.

▶ **Theorem 19.** *Let k, m, u, n be positive integers, and $0 < \varepsilon < \varepsilon'$ where*

$$u = k^{2(1+\varepsilon')cn/\varepsilon'} and m = u^{\varepsilon/(2\log k)}$$

for some universal constant $c > 0$. If there is an algorithm for average-case k-SUM(u, m) that runs in time $T_{\mathsf{kSUM}} = T_{\mathsf{kSUM}}(k, u, m)$, then there is an algorithm for the worst-case $n^{1+\varepsilon'}$-approximate shortest independent vectors problem (SIVP) that runs in time $2^{O(\varepsilon n/\varepsilon' + \log n)} \cdot T_{\mathsf{kSUM}}$.

When we say that a k-SUM algorithm succeeds, we mean that it outputs a k-subset of the input that sums to 0 with probability $1 - \delta$ for some tiny δ. This can be achieved starting from an algorithm that succeeds with (some small) probability p by repeating, at the expense of a multiplicative factor of $1/p \cdot \log(1/\delta)$ in the run-time. We ignore such issues for this exposition, and assume that the algorithm outputs a k-sum with probability $1 - \delta$ for a tiny δ.

Before we proceed to the proof, a few remarks on the parameters of Theorem 19 are in order. First, note that the parameter settings imply that $m^k \gg u$, therefore putting us in the total regime of parameters for k-SUM. Secondly, setting $\varepsilon' = 100$ (say), we get the following consequence: if there is a k-SUM algorithm that, on input $m = u^{\varepsilon/(2\log k)}$ numbers, runs in time roughly m, then we have an n^{101}-approximate SIVP algorithm that runs in time $\approx 2^{\varepsilon cn}$. Now, ε is the "knob" that one can turn to make the SIVP algorithm run faster, assuming a correspondingly fast k-SUM algorithm that works with a correspondingly smaller instance.

Proof. The theorem follows from the following observations:

- First, by Theorem 10, there is a reduction from $\widetilde{O}(\sqrt{n \log m'} \cdot \beta)$-approximate SIVP to $\mathsf{SIS}(m', Q, \beta)$, where we take $m' := \lceil k^{10cn/(k\varepsilon')} n^{10} \rceil$. The reduction produces SIS instances over $\mathbb{Z}_Q$ where $Q \geq (\beta n)^{cn}$ for some constant c, and works as long as the SIS algorithm produces solutions of ℓ_1 norm at most β. If the SIS algorithm runs in time $T_{\mathsf{SIS}} = T_{\mathsf{SIS}}(m', Q, \beta)$, the SIVP algorithm runs in time $(m' + T_{\mathsf{SIS}}) \cdot \mathsf{poly}(n)$. We take $\beta := n^{\varepsilon'}$.
- Second, as our main technical contribution, we show in Lemma 20 how to reduce SIS to k-SUM. Note that Theorem 10 gives us the freedom to pick Q, as long as it is sufficiently large. We will set $Q = q^r$ where $r = \lfloor \varepsilon' \log n / (2 \log k) \rfloor$ for a prime $q \approx u \approx (\beta n)^{cn/r} \approx k^{2(1+\varepsilon')cn/\varepsilon'}$.
  Now, Lemma 20 (with $k = t$) shows a reduction from $\mathsf{SIS}(m', Q, \beta)$ to $2k$-SUM$(\mathbb{Z}_q, m)$ (provided that $m' \gg q^{1/k} k^{4r/k} m^{4/k}$, which holds in this case). The reduction produces a SIS solution with ℓ_1 norm bounded by $k^{2r} \leq \beta$.
  The running time of the resulting algorithm is

$$rm'(m \cdot \mathrm{poly}(k, \log q) + 10 T_{\mathsf{kSUM}}) \approx 2^{O(\varepsilon n/\varepsilon' + \log n)} \cdot T_{\mathsf{kSUM}} .$$

- Finally, by Lemma 16, we know that modular $2k$-SUM over $\mathbb{Z}_q$ can be reduced to k-SUM over the integers in the interval $[-u, u]$ for $u \approx q$ with essentially no overhead.

This finishes the proof. ◀

The following lemma shows our main reduction from SIS to k-SUM. In particular, taking $k = t$, $m' \gg (m^4 q)^{1/k} \cdot (10k)^{4r}$ (so that δ is small), and $m \gg q^{1/k}$ (so that k-SUM$(\mathbb{Z}_q, m)$ is total) gives a roughly rmm'-time reduction from SIS over $\mathbb{Z}_{q^r}$ to k-SUM over $\mathbb{Z}_q$ with high success probability.

▶ **Lemma 20.** *Let m, m', k, r, t be positive integers and $q > (tk)^r$ a prime, and let $Q = q^r$. If there is an algorithm that solves (average-case) k-SUM$(\mathbb{Z}_q, m)$ in time T with success probability p, then there is an algorithm that solves $\mathsf{SIS}(m', Q, \beta)$ in time $r \cdot m'(m \cdot \mathrm{poly}(k, t, \log q) + 10T)/p$ with success probability at least $1 - \delta$ and produces a solution with ℓ_1 norm $\beta \leq (tk)^r$, where*

$$\delta := \frac{100rm(m')^2}{p} \cdot \frac{q^{1/4}}{(m'/(10tk)^{2r+1})^{t/4}} .$$

Proof. At a high level, the idea is to run a variant of the Blum-Kalai-Wasserman [12] algorithm where in each iteration, we call a k-SUM oracle. In particular, on input $a_1, \ldots, a_{m'}$, the algorithm operates as follows.

- In the beginning of the i^{th} iteration, the algorithm starts with a sequence of

  $$m_i := \lceil m'/(10t^2k^2)^{i-1} \rceil$$

  numbers $a_{i,1}, \ldots, a_{i,m_i}$. The invariant is that $a_{i,j} = 0 \pmod{q^{i-1}}$ for all j. It then generates disjoint $S_{i,1}, \ldots, S_{i,m_{i+1}} \subseteq [m_i]$ such that $|S_{i,\ell}| \leq kt$ and $\sum_{j \in S_{i,\ell}} a_{i,j} = 0 \pmod{q^i}$, in a way that we will describe below.

  As the base case, for $i = 1$, $a_{1,j} = a_j$, the input itself, and the invariant is vacuous.
- In the i^{th} iteration, we apply the re-randomization lemma (Corollary 4), computing subsets of t randomly chosen elements from $a_{i,1}, \ldots, a_{i,m_i}$, to generate $m_i^* := 10m\lceil m_{i+1}/p \rceil$ numbers $c_{i,1}, \ldots, c_{i,m_i^*}$.
- Let $d_{i,j} = c_{i,j}/q^{i-1} \pmod{q} \in \mathbb{Z}_q$. Note that this is well-defined because each $c_{i,j} = 0 \pmod{q^{i-1}}$.
- Divide the $d_{i,j}$ into $10\lceil m_{i+1}/p \rceil$ disjoint blocks of m elements each, set $\ell = 1$. For each block, feed the block to the k-SUM algorithm to obtain $d_{i,j_1}, \ldots, d_{i,j_k}$. This yields corresponding subsets $S_1^*, \ldots, S_k^* \in \binom{[m_i]}{t}$ such that $\sum_{j \in S_x^*} a_{i,j}/q^{i-1} = d_{i,j_x} \pmod{q}$. If $d_{i,j_1} + \cdots + d_{i,j_k} = 0 \pmod{q}$ and the sets $S_1^*, \ldots S_k^*, S_{i,1}, \ldots, S_{i,\ell-1}$ are pairwise disjoint, then set $S_{i,\ell} := \bigcup S_x^*$ and increment ℓ.
- If $\ell \leq m_{i+1}$, the algorithm fails. Otherwise, take $a_{i+1,\ell} := \sum_{j \in S_{i,\ell}} a_{i,j}$ for $\ell = 1, \ldots, m_{i+1}$.
- At the end of the r^{th} iteration we obtain a $(kt)^r$-subset of the $a_1, \ldots, a_{m'}$ that sums to 0 $\pmod{Q}$.

We now analyze the correctness, run-time and the quality of output of this reduction.

The reduction calls the k-SUM oracle $\sum m_i^*/m \leq 20rm'/p$ times. The rest of the operations take $rmm'\mathrm{poly}(k, t, \log q)/p$ time for a total of $r \cdot m'(m\mathrm{poly}(k, t, \log q) + 10T)/p$ time, as claimed. Furthermore, the ℓ_1 norm of the solution is $\beta \leq (tk)^r$, as claimed.

Finally, we show that the algorithm succeeds with the claimed probability. Since the sets $S_{i,\ell}$ are disjoint and do not depend on $a_{i,j} - (a_{i,j} \bmod q^i)$, it follows from a simple induction argument that at each step the $a_{i,j}$ are uniformly random and independent elements from $q^{i-1}\mathbb{Z}/q^r\mathbb{Z}$. Therefore, by Corollary 4, the statistical distance of the collection of all $d_{i,j}$ (for a given i) from uniformly random variables that are independent of the $a_{i,j}$ is $\delta_i \leq (m_i^* + 1) \cdot (\frac{qt^t}{m_i^t})^{1/4}$. In total, the statistical distance of all samples from uniform is then at most $\sum \delta_i < \delta/3$ for our choice of parameters. So, up to statistical distance $\delta/3$, we can treat the $d_{i,j}$ as uniformly random and independent elements.

It remains to show that, assuming that the $d_{i,j}$ are uniformly random and independent, then we will find *disjoint* sets $S_{i,1}, \ldots, S_{i,m_{i+1}}$ with $\sum_{j \in S_{i,\ell}} d_{i,j} = 0 \pmod{q}$ at each step except with probability at most $2\delta/3$. Let $\boldsymbol{b}_i := (a_{i,1}/q^{i-1} \bmod q, \ldots, a_{i,m_i}/q^{i-1} \bmod q)$. By Corollary 7, we have

$$p_{\boldsymbol{b}_i, \boldsymbol{d}_i, I, J, t} \leq 10t^2k^2m_{i+1}/m_i \leq 1/2$$

for all $I \in \binom{[m_i]}{\leq v}$ and $J \in \binom{[m_i^*]}{\leq v}$ except with probability at most $10 m_i m_i^* q^{1/4} / \binom{m_i - 1}{t-1}^{1/4} < \delta/3$ for $v := tkm_{i+1} \geq |T|$, $v' := k$, and $\varepsilon := 1/2$.

So, suppose this holds. Notice that $\Pr[d_{i,j_1} + \cdots + d_{i,j_k} = 0 \pmod q] = p$ by definition. And, conditioned on $d_{i,j_1}, \ldots, d_{i,j_k}$, the S_x^* are independent and uniformly random subject to the constraint that $\sum_{j \in S_x^*} a_{i,j}/q^{i-1} = d_{i,j_x} \pmod q$. Therefore, the probability that $S_1^*, \ldots, S_k^*, I := S_{i,1} \cup \cdots \cup S_{i,\ell-1}$ are pairwise disjoint in this case is exactly

$$p_{\boldsymbol{b}_i, \boldsymbol{d}_i, I, J, t} \leq 1/2 \,,$$

where $J := \{j_1, \ldots, j_k\}$. So, each time we call the oracle, we increment ℓ with probability at least $p/2$. It follows from the Chernoff-Hoeffding bound that we increment ℓ at least m_{i+1} times except with probability at most $e^{-m_{i+1}/100} \ll \delta/3$.

Putting everything together, we see that the algorithm fails with probability at most δ, as claimed. ◀

References

1. Amir Abboud, Karl Bringmann, Danny Hermelin, and Dvir Shabtay. SETH-based lower bounds for subset sum and bicriteria path. In *SODA*, 2019.
2. Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *FOCS*, 2014.
3. Divesh Aggarwal and Eldon Chung. A note on the concrete hardness of the shortest independent vector in lattices. *Information Processing Letters*, 167, 2021.
4. Divesh Aggarwal, Jianwei Li, Phong Q. Nguyen, and Noah Stephens-Davidowitz. Slide reduction, revisited—Filling the gaps in SVP approximation. In *CRYPTO*, 2020. URL: `https://arxiv.org/abs/1908.03724`.
5. Miklós Ajtai. Generating hard instances of lattice problems. In *STOC*, 1996.
6. Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of Learning with Errors. *J. Mathematical Cryptology*, 9(3), 2015. URL: `http://eprint.iacr.org/2015/046`.
7. Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange — A new hope. In *USENIX Security Symposium*, 2016.
8. Kyriakos Axiotis, Arturs Backurs, Ce Jin, Christos Tzamos, and Hongxun Wu. Fast modular subset sum using linear sketching. In *SODA*, 2019.
9. Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. Average-case fine-grained hardness. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *STOC*, 2017.
10. Anja Becker, Jean-Sébastien Coron, and Antoine Joux. Improved generic algorithms for hard knapsacks. In *CRYPTO*, 2011.
11. Richard Bellman. Notes on the theory of dynamic programming iv - maximization over discrete sets. *Naval Research Logistics Quarterly*, 3:67–70, 1956. `doi:10.1002/nav.3800030107`.
12. Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM*, 50(4):506–519, 2003. `doi:10.1145/792538.792543`.
13. E. Boix-Adserà, M. Brennan, and G. Bresler. The average-case complexity of counting cliques in Erdős-Rényi hypergraphs. In *FOCS*, 2019.
14. Karl Bringmann. A near-linear pseudopolynomial time algorithm for subset sum. In Philip N. Klein, editor, *SODA*, 2017.
15. Mina Dalirrooyfard, Andrea Lincoln, and V. Vassilevska Williams. New techniques for proving fine-grained average-case hardness. In *FOCS*, 2020.
16. Anka Gajentaan and Mark H Overmars. On a class of $O(n^2)$ problems in computational geometry. *Computational Geometry*, 5(3), 1995. URL: `http://www.sciencedirect.com/science/article/pii/0925772195000222`.

17 Anka Gajentaan and Mark H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Computational Geometry*, 45(4), 2012. URL: `http://www.sciencedirect.com/science/article/pii/S0925772111000927`.

18 Nicolas Gama, Malika Izabachène, Phong Q. Nguyen, and Xiang Xie. Structural lattice reduction: Generalized worst-case to average-case reductions and homomorphic cryptosystems. In Marc Fischlin and Jean-Sébastien Coron, editors, *Eurocrypt*, 2016.

19 Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, 2008. URL: `https://eprint.iacr.org/2007/432`.

20 Oded Goldreich and Guy N. Rothblum. Counting t-cliques: Worst-case to average-case reductions and direct interactive proof systems. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 77–88. IEEE Computer Society, 2018. `doi:10.1109/FOCS.2018.00017`.

21 Oded Goldreich and Guy N. Rothblum. Worst-case to average-case reductions for subclasses of P. In Oded Goldreich, editor, *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. `doi:10.1007/978-3-030-43662-9_15`.

22 Shuichi Hirahara and Nobutaka Shimizu. Nearly optimal average-case complexity of counting bicliques under SETH, 2020. `arXiv:2010.05822`.

23 Ellis Horowitz and Sartaj Sahni. Computing partitions with applications to the knapsack problem. *J. ACM*, 21(2):277–292, 1974. `doi:10.1145/321812.321823`.

24 Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions (extended abstracts). In David S. Johnson, editor, *STOC*, 1989.

25 Ce Jin and Hongxun Wu. A simple near-linear pseudopolynomial time randomized algorithm for subset sum. In *SOSA*, 2019.

26 Paul Kirchner and Pierre-Alain Fouque. Time-memory trade-off for lattice enumeration in a ball, 2016.

27 Konstantinos Koiliaris and Chao Xu. A faster pseudopolynomial time algorithm for subset sum. In Philip N. Klein, editor, *SODA*, 2017.

28 Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Higher lower bounds from the 3SUM conjecture. In *SODA*, 2016.

29 Ravi Kumar and D. Sivakumar. On polynomial-factor approximations to the shortest lattice vector length. *SIAM J. Discrete Math.*, 16(3):422–425, 2003.

30 Daniele Micciancio and Chris Peikert. Hardness of SIS and LWE with small parameters. In *CRYPTO*, 2013.

31 Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. *SIAM Journal of Computing*, 37(1), 2007.

32 NIST. Post-quantum cryptography standardization. URL: `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography`.

33 Mihai Patrascu. Towards polynomial lower bounds for dynamic problems. In *STOC*, 2010.

34 Mihai Patrascu and Ryan Williams. On the possibility of faster SAT algorithms. In *SODA*, 2010.

35 Chris Peikert. A decade of lattice cryptography. *Foundations and Trends in Theoretical Computer Science*, 10(4), 2016.

36 David A. Wagner. A generalized birthday problem. In *CRYPTO*, 2002.

37 Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the International Congress of Mathematicians (ICM 2018)*. 2018.

38 Virginia Vassilevska Williams and Ryan Williams. Finding, minimizing, and counting weighted subgraphs. *SIAM J. Comput.*, 42(3):831–854, 2013. `doi:10.1137/09076619X`.

A Total k-SUM and Computational Geometry

Here, we show that one of the main results in [16, 17] can be extended meaningfully to our setting, i.e., to the case of search k-sum over $\mathbb{Z}_Q$ with $\binom{m}{k} \gg Q$. (In [16, 17], Gajentaan and Overmars only considered decisional 3-SUM.) Specifically, we will reduce the following problem to k-SUM in this regime.

▶ **Definition 21.** *For $d \geq 1$ and $Q, m \geq 2$ with Q prime, the* (Q, m, d)-Plane problem *is the following search problem. The input is $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m \in \mathbb{Z}_Q^{d+1}$. The goal is to find distinct $\boldsymbol{a}_{i_1}, \ldots, \boldsymbol{a}_{i_{d+2}}$ that lie in a d-dimensional affine hyperplane over the field $\mathbb{Z}_Q$. (In other words, $\boldsymbol{a}_{i_{d+1}} - \boldsymbol{a}_{i_{d+2}}$ can be written as a linear combination of $\boldsymbol{a}_{i_1} - \boldsymbol{a}_{i_{d+2}}, \ldots, \boldsymbol{a}_{i_d} - \boldsymbol{a}_{i_{d+2}}$ over $\mathbb{Z}_Q$.)*

▶ **Lemma 22.** *For $d \geq 1$ and $Q, m \geq 2$ with Q prime, there is a reduction from $(d+2)$-SUM$(\mathbb{Z}_Q, m)$ to (Q, m, d)-Plane.*

Proof. Let $f_d : \mathbb{Z}_Q \to \mathbb{Z}_Q^{d+1}$ be the map $f_d(a) := (a, a^2, a^3, \ldots, a^d, a^{d+2})$. E.g., $f_1(a) = (a, a^3)$, $f_2(a) = (a, a^2, a^4)$, etc. On input $a_1, \ldots, a_m \in \mathbb{Z}_Q$, the reduction simply calls its (Q, m, d)-Plane oracle on $f_d(a_1), \ldots, f_d(a_m) \in \mathbb{F}_Q^{d+1}$, receiving as output distinct indices $i_1, \ldots, i_{d+2}$ such that $f_d(a_{i_1}), \ldots, f_d(a_{i_{d+2}})$ lie in a d-dimensional affine hyperplane (assuming that such indices exist). The reduction simply outputs these indices, i.e., it claims that $a_{i_1} + \cdots + a_{i_{d+2}} = 0 \bmod Q$.

Notice that $d+2$ points $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{d+2} \in \mathbb{Z}_Q^{d+1}$ lie in a d-dimensional affine hyperplane if and only if the matrix $(\boldsymbol{b}_1 - \boldsymbol{b}_{d+2}, \boldsymbol{b}_2 - \boldsymbol{b}_{d+2}, \ldots, \boldsymbol{b}_{d+1} - \boldsymbol{b}_{d+2}) \in \mathbb{Z}_Q^{(d+1)\times(d+1)}$ has determinant zero. (Here, we have used the fact that $\mathbb{Z}_Q$ is a field.) So, we consider the matrix

$$\boldsymbol{M} := \boldsymbol{M}(b_1, \ldots, b_{d+2}) := \begin{pmatrix} b_1 - b_{d+2} & b_2 - b_{d+2} & \cdots & b_{d+1} - b_{d+2} \\ b_1^2 - b_{d+2}^2 & b_2^2 - b_{d+2}^2 & \cdots & b_{d+1}^2 - b_{d+2}^2 \\ \vdots & \vdots & \ddots & \vdots \\ b_1^d - b_{d+2}^d & b_2^d - b_{d+2}^d & \cdots & b_{d+1}^d - b_{d+2}^d \\ b_1^{d+2} - b_{d+2}^{d+2} & b_2^{d+2} - b_{d+2}^{d+2} & \cdots & b_{d+1}^{d+2} - b_{d+2}^{d+2} \end{pmatrix} \in \mathbb{F}_Q^{(d+1)\times(d+1)} .$$

We claim that

$$\det(\boldsymbol{M}) = (-1)^d (b_1 + \cdots + b_{d+2}) \cdot \prod_{i<j} (b_j - b_i) ,$$

The result then follows, since this is zero if and only if $b_i = b_j$ for some $i \neq j$ or $b_1 + \cdots + b_{d+2} = 0$. Since by definition the (Q, m, d)-Plane oracle only outputs distinct vectors on a hyperplane, this means that its output must correspond to distinct elements with $a_{i_1} + \cdots + a_{i_{d+2}} = 0 \bmod Q$.

To prove that the determinant has the appropriate form, we first notice that without loss of generality we may take $b_{d+2} = 0$. Next, we define

$$\boldsymbol{M}' := \boldsymbol{M}'(b_1, \ldots, b_{d+1}) := \begin{pmatrix} b_1 & b_2 & \cdots & b_{d+1} \\ b_1^2 & b_2^2 & \cdots & b_{d+1}^2 \\ \vdots & \vdots & \ddots & \vdots \\ b_1^d & b_2^d & \cdots & b_{d+1}^d \\ b_1^{d+1} & b_2^{d+1} & \cdots & b_{d+1}^{d+1} \end{pmatrix} \in \mathbb{F}_Q^{(d+1)\times(d+1)} .$$

This is just a Vandermonde matrix with columns scaled up by b_i. So, its determinant is a scaling of the Vandermonde determinant,

$$\det(\boldsymbol{M}') = b_1 \cdots b_{d+1} \cdot \prod_{i<j} (b_j - b_i) .$$

Finally, we recall Cramer's rule, which in particular tells us that

$$\det(\boldsymbol{M}) = p_{d+1} \det(\boldsymbol{M}')$$

for the unique $\boldsymbol{p} := (p_1, \ldots, p_{d+1}) \in \mathbb{Z}_Q^{d+1}$ satisfying $\boldsymbol{p}^T \boldsymbol{M}' = (b_1^{d+2}, \ldots, b_{d+1}^{d+2})$. I.e., the coordinates of $\boldsymbol{p}$ form the polynomial $p(x) := p_1 + p_2 x + \cdots + p_{d+1} x^d$ such that $p(b_i) = b_i^{d+1}$. The result follows by noting that $p_i = (-1)^{i-1} \sum_{S \in \binom{[d+1]}{d+2-i}} \prod_{j \in S} b_j$. In particular, $p_{d+1} = (-1)^d (b_1 + \cdots + b_{d+1})$, as needed. ◀