Subset Sum in the Absence of Concentration

Per Austrin[1], Petteri Kaski[2], Mikko Koivisto[3], and Jesper Nederlof[4]

1 **School of Computer Science and Communication, KTH Royal Institute of Technology, Sweden**
austrin@csc.kth.se
2 **Helsinki Institute for Information Technology HIIT & Department of Computer Science, Aalto University, Finland**
petteri.kaski@aalto.fi
3 **Helsinki Institute for Information Technology HIIT & Department of Computer Science, University of Helsinki, Finland**
mikko.koivisto@helsinki.fi
4 **Department of Mathematics and Computer Science, Technical University of Eindhoven, The Netherlands**
jespernederlof@hotmail.com

Abstract

We study the exact time complexity of the Subset Sum problem. Our focus is on instances that lack additive structure in the sense that the sums one can form from the subsets of the given integers are not strongly concentrated on any particular integer value. We present a randomized algorithm that runs in $O\big(2^{0.3399n}B^4\big)$ time on instances with the property that no value can arise as a sum of more than B different subsets of the n given integers.

1998 ACM Subject Classification F.2.1 Numerical Algorithms and Problems, F.2.2 Nonnumerical Algorithms and Problems, G.2.1 Combinatorics

Keywords and phrases subset sum, additive combinatorics, exponential-time algorithm, homomorphic hashing, Littlewood–Offord problem

Digital Object Identifier 10.4230/LIPIcs.STACS.2015.48

1 Introduction

Given integers $a_1, a_2, \ldots, a_n \in \mathbb{Z}$ and a target integer $t \in \mathbb{Z}$ as input, the NP-complete Subset Sum problem asks whether there exists a subset $S \subseteq [n]$ with $\sum_{j\in S} a_j = t$. Despite the apparent simplicity of the problem statement, to date there has been modest progress on exact algorithms [11] for Subset Sum. Indeed, from a worst-case performance perspective the fastest known algorithm runs in $O^*(2^{n/2})$ time[1] and dates back to the 1974 work of Horowitz and Sahni [13]. Improving the worst-case running time is a well-established open problem [28, §53].

Improved algorithms are known in special cases where one assumes control on the additive structure in the *steps*[2] $a_1, a_2, \ldots, a_n$. In particular this is the case if we assume

[1] The $O^*(\cdot)$ notation suppresses a multiplicative factor polynomial in the input size.

[2] The term *step* originates from the study of the Littlewood–Offord problem in additive combinatorics, see e.g. Tao and Vu [26] and [25, §7]. In this context the integers $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ define the step-lengths of an n-step random walk on $\mathbb{Z}$ given by the random variable $S(\mathbf{a}) = \sum_{i=1}^{n} a_i\epsilon_i$, where the values $\epsilon_1, \epsilon_2, \ldots, \epsilon_n \in \{-1, 1\}$ are selected independently and uniformly at random. (The Subset

32nd Symposium on Theoretical Aspects of Computer Science (STACS 2015).
Editors: Ernst W. Mayr and Nicolas Ollinger; pp. 48–61
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

that the possible sums that one can form out of the steps are supported by a small set of values. Assuming such additive structure is available, a number of algorithmic techniques exist to improve upon the Horowitz–Sahni bound, ranging from Bellman's classical dynamic programming [5] to algebraization [16, 18] and to parameterized techniques [9].

Given that additive structure enables improved algorithms, it would thus be natural to expect, *a priori*, that the worst-case instances are the ones that lack any additive structure. In a cryptographic context such instances are frequently assumed to be *random*. Yet, recently Howgrave-Graham and Joux [14] made a breakthrough by showing that random instances can in fact be solved in $O^*(2^{0.337n})$ time; a correct derivation of this bound and an improvement to $O^*(2^{0.291n})$ are given by Becker, Coron, and Joux [4].

The results of Joux *et al.* raise a number of intriguing combinatorial and algorithmic questions. Given that improved algorithms exist for random instances, precisely what types of hard instances remain from a worst-case analysis perspective? What are the *combinatorial* properties that the hard instances must have? Conversely, from an algorithms standpoint one would like to narrow down precisely what *intrinsic* property of the steps $a_1, a_2, \ldots, a_n$ enables algorithm designs such as the Joux *et al.* results. Such a quest for intrinsic control is further motivated by the relatively recent identification of a large number of natural dichotomies between *structure* and *randomness* in combinatorics (cf. Tao [22, 23, 24], Trevisan [27], and Bibak [6]). Do there exist algorithm designs that capitalize on *pseudorandomness* (absence of structure) to improve over the worst-case performance?

In this paper we seek to take algorithmic advantage of the *absence* of additive structure in instances of Subset Sum. A prerequisite to such a goal is to have a combinatorial parameter that measures the extent of additive structure in an instance. The close relationship between the Subset Sum problem and the *Littlewood–Offord problem* in additive combinatorics [25, §7] suggests that one should study measures of additive structure in the context of the latter.

We show that such transfer is indeed possible, and leads to improved exact algorithms for pseudorandom instances of Subset Sum. Our measure of choice for pseudorandomness is the *concentration probability* employed, for example, by Tao and Vu [26] in the context of inverse theorems in Littlewood–Offord theory. We use an equivalent but fully combinatorial definition that will be more convenient in the context of Subset Sum. Define the set function $a : 2^{[n]} \to \mathbb{Z}$ for all $X \subseteq [n]$ by $a(X) = \sum_{j \in X} a_j$. We now assume that we have a uniform upper bound on the size of the preimages $a^{-1}(u) = \{X \subseteq [n] : a(X) = u\}$ for $u \in \mathbb{Z}$. For $B \geq 1$ we say that the instance $a_1, a_2, \ldots, a_n$ has *B-bounded concentration* if for all $u \in \mathbb{Z}$ it holds that $|a^{-1}(u)| \leq B$.[3] That is, no value $u \in \mathbb{Z}$ may occur as a sum $a(X) = u$ for more than B different subsets $X \subseteq [n]$. The extreme case $B = 1$ captures the notion of *additive independence* or *dissociativity* in additive combinatorics [25, §4.32]. The other extreme case $B = 2^n$ is achieved by $a_1 = a_2 = \ldots = a_n = 0$, and more generally, an instance of high density (see §1.2) automatically has high concentration.

Our main result is that all instances *without* strong additive structure (without exponential concentration of sums) can be solved faster than the Horowitz–Sahni time bound $O^*(2^{n/2})$. A quantitative claim is as follows.[4]

Sum problem is equivalent to asking whether the outcome $S(\mathbf{a}) = s$ has positive probability for $s = 2t - \sum_{j=1}^{n} a_j$.)

[3] In terms of the Littlewood–Offord random walk, B-bounded concentration is equivalent to the assertion that for all $u \in \mathbb{Z}$ the probability of the outcome $S(\mathbf{a}) = u$ is at most $B/2^n$.

[4] The running time versus the concentration bound in Theorem 1 can be sharpened somewhat; our subsequent analysis enables a more precise smooth tradeoff curve between the concentration bound B and the running time of the algorithm captured by Equation (5) in what follows. When the instance

▶ **Theorem 1.** *There exists a randomized algorithm for* Subset Sum *that with probability* $1 - o(1)$ *solves instances with* B*-bounded concentration in time* $O^*\big(2^{0.3399n}B^4\big)$.

Theorem 1 shows that we can afford quite considerable additive structure (exponential concentration of sums) in an instance, and still remain below the Horowitz–Sahni $O^*(2^{n/2})$ worst-case upper bound. Here we should note that we do not show how to *decide* whether a given instance has B-bounded concentration. Yet, by invoking our algorithm with $B = 1, 2, 4, \ldots$ we can conclude within the time bound in Theorem 1 that we have decided the instance correctly with high probability *or* the instance does not have B-bounded concentration. Thus, we are left with a dichotomy of possibilities from the perspective of worst-case algorithm design:

> *Either* worst-case instances *must* have considerable additive structure (and the Horowitz–Sahni bound thus remains uncontested), *or* Subset Sum has improved worst-case algorithms.

1.1 Methodology and Contributions

Our main challenge is to show that having control *only* on the concentration parameter B enables algorithmic implications. Here our approach is to alleviate the randomness assumptions on one of the aforementioned Joux *et al.* algorithm designs by injecting extra entropy into the algorithm.

In particular, we view the choice of the moduli in the algorithm as defining a *family of hash functions*. We show that control only on B suffices for probabilistic control on the extent to which a fixed solution is witnessed in the codomain of a *random* hash function. Thus, we obtain an algorithm design by (i) selecting a random hash function, (ii) selecting a random element in the codomain, and (iii) searching the preimage for solutions.

The Howgrave-Graham–Joux two-level design. To highlight our contribution in more detail, let us begin with a high-level review of a Howgrave-Graham–Joux [14] design, adapted from the careful presentation by Becker [3, §4.2], that we then proceed to analyze and instrument in what follows. The algorithm searches for a solution S by building the solution from four disjoint parts

$$S = S_1 \cup S_2 \cup S_3 \cup S_4 \quad \text{with} \quad |S_1| = |S_2| = |S_3| = |S_4| = |S|/4\,. \tag{1}$$

The search is executed along what can be viewed as a two-level recursion tree, which is best analyzed from a top-down perspective. (The actual algorithm will operate in the reverse direction from the bottom up when searching for S.) The first level splits S into two halves T and $S \setminus T$ for some $T \in \binom{S}{|S|/2}$. The second level then splits T and $S \setminus T$ into halves. That is, T splits to L and $T \setminus L$ for some $L \in \binom{T}{|S|/4}$, and $S \setminus T$ splits to R and $(S \setminus T) \setminus R$ for some $R \in \binom{S\setminus T}{|S|/4}$. The triple (T, L, R) now naturally splits S into four parts

$$S_1 = L\,, \quad S_2 = T \setminus L\,, \quad S_3 = R\,, \quad S_4 = (S \setminus T) \setminus R\,. \tag{2}$$

The advantage of the tree-like design over a flat design (1) is that subtrees at T and $S \setminus T$ are (essentially) independent and the T-component of the three-tuple (T, L, R) gives direct control on what happens at nodes T and $S \setminus T$ in the tree. The design now controls the size of

---

does not have B-bounded concentration, the algorithm will run in the claimed time in the parameters n and B, but in this case the algorithm is not guaranteed to find a solution with high probability.

the search space by fixing two $|S|/2$-bit coprime moduli, M_1 and M_2, and then *constraining uniformly at random* the congruence classes of the values $a(T)$, $a(L)$, and $a(R)$, which are constrained modulo M_1M_2, M_1, and M_1, respectively.[5] A clever application of a theorem of Nguyen, Shparlinski, and Stern [19, Theorem 3.2] on the distribution of modular sums across the ensemble of all instances $a_1, a_2, \ldots, a_n$ then enables Howgrave-Graham and Joux to control on the size of the search space on all but an exponentially negligible fraction of instances to obtain $O^*(2^{0.337n})$ running time on random instances. This is the starting point of our present work.

Our contributions. In contrast to the Howgrave-Graham–Joux analysis, our objective is to *localize* the ensemble analysis to individual instances with control only on the instance-specific additive structure captured by B. Our central observation is that we can obtain further probabilistic control on the algorithm by studying how the selection of the moduli M_1 and M_2 affects the search space. In particular it makes sense to treat the selection of M_1 and M_2 as the task of selecting a hash function from a formal family of hash functions h_{M_1,M_2} with

$$h_{M_1,M_2}(T,L,R) = \big(a(T) \bmod M_1M_2,\, a(L) \bmod M_1,\, a(R) \bmod M_1\big) \in \mathbb{Z}_{M_1M_2} \times \mathbb{Z}_{M_1} \times \mathbb{Z}_{M_1}\,.$$

Our main contribution is now that, assuming control only on B, and letting $M_1 = p$ and $M_2 = q$ to be distinct random primes, we find that any fixed solution S is extensively witnessed in the codomain $\mathbb{Z}_{pq} \times \mathbb{Z}_p \times \mathbb{Z}_p$ of a random $h_{p,q}$ (Lemma 3). Furthermore, we show that the size of the search space traversed by (a minor instrumentation of) the two-level Howgrave-Graham–Joux design is bounded with essentially the same running time guarantee (Lemma 4).[6] Here we stress that our main contribution is analytical (Lemma 3) and in explicating the relevance of the hash family that enables us to inject entropy into the algorithm to cope with control on B only. In essence, our contribution is in localizing the outliers substantially deviating from the ensemble average (Nguyen *et al.* [19, Theorem 3.2]) to instances with substantial additive structure (exponential concentration of sums).

1.2 Related Work

The complexity of a Subset Sum instance is commonly measured by the *density*, defined as $n/(\log_2 \max_i a_i)$. From the perspective of worst-case guarantees, the Schroeppel–Shamir algorithm [21] that uses $O^*(2^{n/2})$ time and $O^*(2^{n/4})$ space remains uncontested for instances of density at most 2. However, improved space–time tradeoffs have been discovered recently first for random instances by Dinur, Dunkelman, Keller, and Shamir [8], and then generalized to worst-case instances by Austrin, Kaski, Koivisto, and Määttä [2]. For worst-case instances with density greater than 2, Bellman's classical dynamic programming algorithm [5] remains the fastest. If allowed only polynomial space, improving the $O^*(2^n)$-time exhaustive search

[5] Because the set function a is modular and a solution satisfies $a(S) = t$, control on T, L, R gives control on the corresponding right siblings $S \setminus T$, $T \setminus L$, $(S \setminus T) \setminus R$ and hence gives control on the entire search.

[6] Observe that our bound $O^*(2^{0.3399n}B^4)$ is worse than the $O^*(2^{0.337n})$ obtainable for random instances with a corrected version of the original Howgrave-Graham and Joux [14] design (as described by Becker, Coron, and Joux [4] and Becker [3, §4.2]). This results from the fact that we analyze only the *two-level* design relative to control on B, whereas $O^*(2^{0.337n})$ time would require the analysis of a more intricate three-level design for a gain in the third decimal digit of the exponent of the running time. Our contribution in the present work should be viewed as not optimizing the base of the exponential via increasingly careful constructions but rather showing that control on B alone is sufficient to go below the Horowitz–Sahni bound by a slightly more refined analysis of the Howgrave-Graham–Joux design.

algorithm for density at most 1 is an open problem, while for density $1 + \Theta(1)$ there is an improved algorithm by Lokshtanov and Nederlof [18].

Impagliazzo and Naor [15] show that, with respect to polynomial time solvability, random instances are the hardest when the density is close to 1. As already mentioned, for such instances the $O^*(2^{n/2})$ time bound was recently improved to $O^*(2^{0.337n})$ [14] and subsequently to $O^*(2^{0.291n})$ [4, 3]. Interestingly, almost all instances of density at most 0.94 can be reduced to the problem of finding shortest non-zero vectors in lattices (cf. Lagarias and Odlyzko [17] and Coster *et al.* [7]). Flaxman and Przydatek's algorithm [10] solves random instances in expected polynomial time if the density is $\Omega(n/\log_2 n)$, or equivalently, $\log_2 \max_i a_i = O((\log_2 n)^2)$.

Random instances have been studied also for other NP-hard problems; see, for example, Achlioptas's survey on random satisfiability [1]. We are not aware of prior work on exact algorithms that make use of pseudorandomness (*absence* of structure) and improve over the worst case in such an explicit way as we do in the present work.

2 Preliminaries

This section makes a review of standard notation and results used in this paper.

We write $[n]$ for the set $\{1, 2, \ldots, n\}$. For a finite set U, we write 2^U for the set of all subsets of U and $\binom{U}{k}$ for the set of all subsets of U of size k. For $0 \leq \sigma \leq 1$ let $H(\sigma) = -\sigma \log_2 \sigma - (1-\sigma)\log_2(1-\sigma)$ be the binary entropy function with $H(0) = H(1) = 0$. For all integers $n \geq 1$ and $0 \leq \sigma \leq 1$ such that σn is an integer, we have by Stirling's formula [20] that $\binom{n}{\sigma n} \leq 2^{nH(\sigma)}$.

We write $\mathbb{Z}$ for the set of integers and $\mathbb{Z}_{\geq 1}$ for the set of all positive integers. We will need the following weak version of the Prime Number Theorem [12, p. 494, Eq. (22.19.3)].

▶ **Lemma 2.** *For all large enough integers b it holds that there exist at least $2^b/b$ prime numbers p in the interval $2^b < p < 2^{b+1}$.*

For a modulus $M \in \mathbb{Z}_{\geq 1}$ and $x, y \in \mathbb{Z}$, we write $x \equiv y \pmod M$, or $x \equiv_M y$ for short, to indicate that M divides $x - y$.

For a logical proposition P, we write $[P]$ to indicate a 1 if P is true and a 0 if P is false.

3 The Algorithm

This section proves Theorem 1. Suppose we are given an instance $a_1, a_2, \ldots, a_n, t \in \mathbb{Z}$ as input. We assume that the instance has B-bounded concentration.

3.1 Preprocessing and Parameters of the Input

By resorting to routine randomized preprocessing (detailed in the full version) and then invoking the main algorithm (to be described) a polynomial (in the original input size) number of times, we may assume that the input to the main algorithm has the following structure:

(a) the input $a_1, a_2, \ldots, a_n, t$ consists of positive integers only,
(b) $a_1 + a_2 + \ldots + a_n + t \leq 2^{\tau n}$ for $\tau > 0$ a constant independent of n,
(c) the solution $S \subseteq [n]$, if any, has size $s = |S|$ *that is known to us*,
(d) $n/100 \leq s \leq n/2$,
(e) both n and s are multiples of 8, and
(f) the instance has B-bounded concentration.

Define $\sigma = s/n$. In particular, $1/100 \leq \sigma \leq 1/2$.

3.2 The Hash Functions

We are interested in discovering the set S (if such a set exists) by assembling it from four equally-sized disjoint parts. Recalling our discussion in §1.1, we will follow a tree-based design with domain

$$\mathscr{D}(S) = \left\{(T, L, R) : T \in \tbinom{S}{s/2}, L \in \tbinom{T}{s/4}, R \in \tbinom{S\setminus T}{s/4}\right\}.$$

Since we are always analyzing a fixed arbitrary solution S, we suppress S and simply write $\mathscr{D}$. The following family of hash functions seeks to witness at least one split from $\mathscr{D}$ with high probability. Towards this end, for $p, q \in \mathbb{Z}_{\geq 1}$ define the function

$$h_{p,q} : \mathscr{D} \to \mathbb{Z}_{pq} \times \mathbb{Z}_p \times \mathbb{Z}_p$$

for all $(T, L, R) \in \mathscr{D}$ by

$$h_{p,q}(T, L, R) = \big(a(T) \bmod pq,\ a(L) \bmod p,\ a(R) \bmod p\big). \tag{3}$$

Our main lemma shows that $\mathscr{D}$ is indeed extensively witnessed in the codomain $\mathbb{Z}_{pq} \times \mathbb{Z}_p \times \mathbb{Z}_p$. The sizes of p and q will be judiciously chosen as follows. Let $p_* = \binom{s/2}{s/4}/B$ and $q_* = \binom{s}{s/2}/(Bp_*)$.[7] Let $\lambda = \log(p_*)/n \approx \sigma/2 - \log(B)/n$.

▶ **Lemma 3.** *Let p and q be independently chosen random primes in the range $[p_*, 2p_*]$ and $[q_*, 2q_*]$, respectively. Then with probability at least $1/2$, it holds that $|h_{p,q}(\mathscr{D})| \geq 2^{-(19+3\tau/\lambda)} s^{-6} p^3 q$.*

Observe in particular that the codomain has size $p^3 q$. We will prove Lemma 3 in §4.

3.3 The Search Subroutine

Once p and q have been fixed, we need a compatible search subroutine that carries out the search in a random preimage of $h_{p,q}^{-1}$. Towards this end, let us assume that p and q are fixed and coprime.

The high-level structure of the search subroutine is captured in the following lemma.

▶ **Lemma 4.** *Suppose that $k_{\mathrm{T}} \in \mathbb{Z}_{pq}$, $k_{\mathrm{L}} \in \mathbb{Z}_p$, and $k_{\mathrm{R}} \in \mathbb{Z}_p$ have been chosen independently and uniformly at random. Then, there exists a randomized algorithm that searches for subsets $S \in \binom{[n]}{s}$ such that both of the following requirements hold:*

(i) *$a(S) = t$; and*

(ii) *there exist sets $T \in \binom{S}{s/2}$, $L \in \binom{S}{s/4}$, and $R \in \binom{S\setminus T}{s/4}$ such that we have*

$$a(T) \equiv k_{\mathrm{T}} \pmod{pq}, \quad a(L) \equiv k_{\mathrm{L}} \pmod{p}, \quad \text{and} \quad a(R) \equiv k_{\mathrm{R}} \pmod{p}.$$

For every choice of $k_{\mathrm{T}}, k_{\mathrm{L}}, k_{\mathrm{R}}$, and every subset S satisfying these conditions, the algorithm finds S with probability at least $1/n^2$ (over the internal randomness of the algorithm).

The expected running time of the algorithm over the choice of $(k_{\mathrm{T}}, k_{\mathrm{L}}, k_{\mathrm{R}})$ is

$$O\big(n^2 \cdot 2^{\frac{1}{2}H(\sigma/4)} + n^2 \cdot 2^{H(\sigma/4)n}/p + n^2 B \cdot 2^{2H(\sigma/4)n}/(p^2 q)\big). \tag{4}$$

The running time bound holds uniformly for all coprime choices of p and q.

The proof of this lemma is given in §5.

[7] For completeness, we round p_* to 1 if $p_* < 1$. However, in this case the running time becomes $\Omega\big(\binom{s/2}{s/4}^4\big)$, rendering the algorithm slow and uninteresting.

3.4 Completing the Proof of Theorem 1

We now combine Lemma 3 and Lemma 4 to yield the algorithm design for Theorem 1.

The algorithm starts by selecting a random independent pair p, q of primes with $p \in [p_*, 2p_*]$ and $q \in [q_*, 2q_*]$ and then selects uniform and independent $k_{\mathrm{T}} \in \mathbb{Z}_{pq}$, $k_{\mathrm{L}} \in \mathbb{Z}_p$, $k_{\mathrm{R}} \in \mathbb{Z}_p$. Then we run the algorithm of Lemma 4 with these parameters. If we find a solution, the algorithm reports that solution. Otherwise the algorithm reports that the instance has no solution.

Let us now analyze the success probability and running time of the algorithm. The output of the algorithm is always correct if the instance has no solution, so let us assume that the instance has a solution S. Let $\alpha = 2^{-(19+3\tau/\lambda)}s^{-6}$, and note that α is inversely polynomial in the input size. From Lemma 3 we have that with probability at least $1/2$ we obtain a pair of primes p, q such that $|h_{p,q}(\mathscr{D})| \geq \alpha p^3 q$. Conditioning on this event, we have that $(k_{\mathrm{T}}, k_{\mathrm{L}}, k_{\mathrm{R}}) \in h_{p,q}(\mathscr{D})$ with probability at least α. Conditioned on $(k_{\mathrm{T}}, k_{\mathrm{L}}, k_{\mathrm{R}}) \in h_{p,q}(\mathscr{D})$, the algorithm of Lemma 4 finds the solution S with probability at least $1/n^2$. In total, the probability that the algorithm finds a solution S is at least $\alpha' := \alpha/(2n^2)$.

The algorithm runs in expected time

$$T = n^2 \cdot O\big(2^{\frac{1}{2}H(\sigma/4)n} + 2^{H(\sigma/4)n}/p + B \cdot 2^{2H(\sigma/4)n}/(p^2 q)\big)\,.$$

By Markov's inequality, with probability at most $\alpha'/2$ it runs in time at most $2T/\alpha'$, so even if we terminate it after $2T/\alpha'$ steps, it still has an $\alpha'/2$ chance of finding S. To amplify this to $1 - o(1)$ we repeat the whole procedure n/α' times, and the overall running time for the algorithm is (suppressing terms polynomial in n)

$$\begin{aligned} 2nT/(\alpha')^2 &= O^*\big(2^{\frac{1}{2}H(\sigma/4)n} + 2^{H(\sigma/4)n}/p + B \cdot 2^{2H(\sigma/4)n}/(p^2 q)\big) \\ &= O^*\big(2^{\frac{1}{2}H(\sigma/4)n} + B2^{(H(\sigma/4)-\sigma/2)n} + B^4 \cdot 2^{(2H(\sigma/4)-3\sigma/2)n}\big), \end{aligned} \tag{5}$$

where we used that $p \geq p_* = \binom{s/2}{s/4}/B \geq 2^{s/2}/(Bs)$ and $q \geq q_* = \binom{s}{s/2}/(Bp_*) \geq 2^{s/2}/(Bs)$.

The first two summands in (5) are maximized in the range $0 \leq \sigma \leq 1/2$ when $\sigma = 1/2$ and are both bounded by $B2^{0.3n}$. The third summand is maximized when $\sigma = 4/9$ where it is roughly $B^4 2^{0.3399n}$, giving the claimed time bound in Theorem 1. The proof of Theorem 1 is now complete.

4 Analysis of the Hash Function

This section proves Lemma 3.

4.1 Size of the Image Under Bounded Collisions

Our first objective is to bound the size of the image $h_{p,q}(\mathscr{D})$ from below subject to the assumption that the parameters p, q have "few collisions" in a sense to be made precise under the assumptions in Lemma 5.

For a subset $X \subseteq S$, a modulus $M \in \mathbb{Z}_{\geq 1}$, and $k \in \mathbb{Z}_M$, let us count the number of halves of X that land in the congruence class of k modulo M by

$$f_{M,k}(X) = \big|\big\{Y \in \tbinom{X}{|X|/2} : a(Y) \equiv k \pmod{M}\big\}\big|\,. \tag{6}$$

Let us say that a set $X \subseteq S$ is γ-*well-spread* relative to a modulus $M \in \mathbb{Z}$ if

$$C_M(X) = \sum_{k \in \mathbb{Z}_M} f_{M,k}(X)^2 \leq \gamma \frac{\binom{|X|}{|X|/2}^2}{M}\,. \tag{7}$$

Note that if $a(Y)$ over $Y \in \binom{X}{|X|/2}$ is evenly distributed over all M modular classes, we would have $C_M(X) = \binom{s}{s/2}^2/M$.

▶ **Lemma 5.** *Let $p, q \in \mathbb{Z}_{\geq 1}$ be fixed so that*

(i) *S is γ-well-spread relative to the modulus pq, and*

(ii) *for at least half of all $T \in \binom{S}{s/2}$, it holds that both T and $S \setminus T$ are γ-well-spread relative to the modulus p.*

Then $|h_{p,q}(\mathscr{D})| \geq p^3 q/(2\gamma^3)$.

Proof. Let $\mathcal{T}_p$ consist of all $T \in \binom{S}{s/2}$ such that both T and $S \setminus T$ are γ-well-spread relative to p. Let us write $\mathscr{D}_p$ for the subfamily of $\mathscr{D}$ consisting of all triples $(T, L, R) \in \mathscr{D}$ such that $T \in \mathcal{T}_p$. We thus have by assumption (ii) that

$$|\mathscr{D}_p| \geq \frac{1}{2}|\mathscr{D}| = \frac{1}{2}\binom{s}{s/2}\binom{s/2}{s/4}^2 . \tag{8}$$

It suffices to establish the conclusion for $m = |h_{p,q}(\mathscr{D}_p)|$. Towards this end, let us analyze collisions of $h_{p,q}$ on $\mathscr{D}_p$. Let $\mathscr{C}_p$ consist of all pairs $(T_1, L_1, R_1), (T_2, L_2, R_2) \in \mathscr{D}_p$ with $h_{p,q}(T_1, L_1, R_1) = h_{p,q}(T_2, L_2, R_2)$.

We start with a routine quadratic bound. Let $c_1, c_2, \ldots, c_m$ be the sizes of preimages of $h_{p,q}$ on $\mathscr{D}_p$. We have $|\mathscr{C}_p| = \sum_{i=1}^m c_i^2$ and $|\mathscr{D}_p| = \sum_{i=1}^m c_i$. By the Cauchy–Schwarz inequality we thus have

$$|h_{p,q}(\mathscr{D}_p)| = m \geq |\mathscr{D}_p|^2/|\mathscr{C}_p| . \tag{9}$$

The claim thus follows by (8) and (9) if we can obtain sufficient control on $|\mathscr{C}_p|$. Recalling (3) and (6), we have

$$\begin{aligned}
|\mathscr{C}_p| &= \sum_{\substack{T_1 \in \mathcal{T}_p \\ T_2 \in \mathcal{T}_p}} [a(T_1) \equiv_{pq} a(T_2)] \sum_{\substack{L_1 \in \binom{T_1}{s/4} \\ L_2 \in \binom{T_2}{s/4}}} [a(L_1) \equiv_p a(L_2)] \sum_{\substack{R_1 \in \binom{S \setminus T_1}{s/4} \\ R_2 \in \binom{S \setminus T_2}{s/4}}} [a(R_1) \equiv_p a(R_2)] \\
&= \sum_{\substack{T_1 \in \mathcal{T}_p \\ T_2 \in \mathcal{T}_p}} [a(T_1) \equiv_{pq} a(T_2)] \sum_{\ell \in \mathbb{Z}_p} f_{p,\ell}(T_1) f_{p,\ell}(T_2) \sum_{\ell \in \mathbb{Z}_p} f_{p,\ell}(S \setminus T_1) f_{p,\ell}(S \setminus T_2) \\
&\leq \underbrace{\sum_{\substack{T_1 \in \mathcal{T}_p \\ T_2 \in \mathcal{T}_p}} [a(T_1) \equiv_{pq} a(T_2)]}_{\text{(a)}} \underbrace{\max_{\substack{T_1 \in \mathcal{T}_p \\ T_2 \in \mathcal{T}_p}} \sum_{\ell \in \mathbb{Z}_p} f_{p,\ell}(T_1) f_{p,\ell}(T_2)}_{\text{(b)}} \underbrace{\max_{\substack{T_1 \in \mathcal{T}_p \\ T_2 \in \mathcal{T}_p}} \sum_{\ell \in \mathbb{Z}_p} f_{p,\ell}(S \setminus T_1) f_{p,\ell}(S \setminus T_2)}_{\text{(c)}} .
\end{aligned}$$

Using (6) we can bound (a) from above by

$$\sum_{\substack{T_1 \in \mathcal{T}_p \\ T_2 \in \mathcal{T}_p}} [a(T_1) \equiv_{pq} a(T_2)] \leq \sum_{k \in \mathbb{Z}_{pq}} f_{pq,k}(S)^2 = C_{pq}(S) .$$

Using the Cauchy–Schwarz inequality we can bound (b) from above by

$$\max_{\substack{T_1 \in \mathcal{T}_p \\ T_2 \in \mathcal{T}_p}} \sum_{\ell \in \mathbb{Z}_p} f_{p,\ell}(T_1) f_{p,\ell}(T_2) \leq \max_{T \in \mathcal{T}_p} \sum_{\ell \in \mathbb{Z}_p} f_{p,\ell}(T)^2 = \max_{T \in \mathcal{T}_p} C_p(T) .$$

This bound on (b) applies also to (c) because we have $T \in \mathcal{T}_p$ if and only if $S \setminus T \in \mathcal{T}_p$. Combining the bounds on (a)–(c) and then using the conditions of S and every $T \in \mathcal{T}_p$ being

γ-well-spread, we conclude that

$$|\mathscr{C}_p| \leq C_{pq}(S) \cdot \left(\max_{T \in \mathcal{T}_p} C_p(T)\right)^2 \leq \gamma \frac{\binom{s}{s/2}}{pq} \left(\gamma \frac{\binom{s/2}{s/4}}{p}\right)^2 = \frac{\gamma^3 |\mathscr{D}|}{p^3 q}\,. \tag{10}$$

The lemma follows now follows from (8), (9), and (10). ◀

4.2 Bounded Collisions Happen with High Probability

This section shows that well-spread moduli are a high-probability event for $p, q \in \mathbb{Z}_{\geq 1}$ selected from an appropriate random ensemble.

Besides control on collisions over a modulus (7), we require control on collisions over the integers. This control is available via bounded concentration. For all $X \subseteq S$ define

$$C_\infty(X) = \sum_{Y_1, Y_2 \in \binom{X}{|X|/2}} [a(Y_1) = a(Y_2)]\,. \tag{11}$$

▶ **Lemma 6.** *For all $X \subseteq S$ we have $C_\infty(X) \leq \binom{|X|}{|X|/2} B$.*

Proof. From (11) we have that there are at most $\binom{|X|}{|X|/2}$ ways to select Y_1. By B-bounded concentration, there are at most B ways to select an $Y_2 \in a^{-1}(a(Y_1))$. ◀

For the next lemma, we recall the parameter τ from §3.1, satisfying $a_1 + \ldots + a_n + t \leq 2^{\tau n}$.

▶ **Lemma 7.** *Let $\mathscr{M}$ be a nonempty set of integers each of which has all prime factors at least $2^{\lambda n}$ for some $\lambda > 0$. Suppose that we select an $M \in \mathscr{M}$ uniformly at random. Then, for any $T \subseteq S$ it holds that*

$$\mathbb{E}_{M \in \mathscr{M}}\big[C_M(T)\big] \leq \binom{|T|}{|T|/2} B + \frac{\binom{|T|}{|T|/2}^2}{|\mathscr{M}|} \cdot 2^{\tau/\lambda}\,.$$

Proof. Fix a nonempty $T \subseteq S$. For an arbitrary $M \in \mathscr{M}$ we have from (7) and (6) that

$$C_M(T) = \sum_{L_1, L_2 \in \binom{T}{|T|/2}} [a(L_1) \equiv_M a(L_2)] = \sum_{L_1, L_2 \in \binom{T}{|T|/2}} [M \text{ divides } a(L_1) - a(L_2)]\,.$$

By linearity of expectation thus

$$\mathbb{E}_{M \in \mathscr{M}}\big[C_M(T)\big] = \sum_{L_1, L_2 \in \binom{T}{|T|/2}} \Pr_{M \in \mathscr{M}}\big[M \text{ divides } a(L_1) - a(L_2)\big]\,. \tag{12}$$

The terms in the sum (12) split into two cases. If $a(L_1) = a(L_2)$, then M divides $a(L_1) - a(L_2)$ with probability 1. From (11) we observe that the sum of these terms is exactly $C_\infty(T)$. Apply Lemma 6 to bound $C_\infty(T)$ from above. If $a(L_1) \neq a(L_2)$, then we observe that, by virtue of preprocessing, $|a(L_1) - a(L_2)| \leq 2^{\tau n}$. This implies that $a(L_1) - a(L_2)$ has at most τ/λ prime factors (with repetition) of size at least $2^{\lambda n}$. Any M that divides $a(L_1) - a(L_2)$ must be created from these τ/λ factors and hence there are at most $2^{\tau/\lambda}$ possible values for M. ◀

For the next lemma, recall that $p_* = \binom{s/2}{s/4}/B$ and $q_* = \binom{s}{s/2}/(Bp_*)$.

▶ **Lemma 8.** *With probability at least $1/2$, a random independent pair (p, q) of primes from $[p_*, 2p_*] \times [q_*, 2q_*]$ satisfies the assumptions of Lemma 5 with $\gamma = 2^{6+\tau/\lambda} s^2$, where $\lambda = \frac{\log p_*}{n}$.*

Proof. Let us start with assumption (ii) in Lemma 5. Take $\mathscr{M}$ to be the set of primes p in the interval $[p_*, 2p_*]$. Define $Z(p) = \mathbb{E}_{T\in\binom{S}{s/2}}[C_p(T)]$. We then have

$$\mathbb{E}_{p\in\mathscr{M}}[Z(p)] = \mathbb{E}_{p\in\mathscr{M}}\left[\mathbb{E}_{T\in\binom{S}{s/2}}[C_p(T)]\right] = \mathbb{E}_{T\in\binom{S}{s/2}}\left[\mathbb{E}_{p\in\mathscr{M}}[C_p(T)]\right]$$
$$\leq \binom{s/2}{s/4}B + \frac{\binom{s/2}{s/4}^2}{|\mathscr{M}|}\cdot 2^{\tau/\lambda} = \binom{s/2}{s/4}^2\left(\frac{1}{p_*} + \frac{2^{\tau/\lambda}}{|\mathscr{M}|}\right),$$

where the inequality is Lemma 7. Applying Markov's inequality and $|\mathscr{M}| \geq p_*/\log(p_*)$ (due to Lemma 2), we conclude that with probability $\geq 3/4$ over a random $p \in \mathscr{M}$ it holds that

$$Z(p) \leq 4\binom{s/2}{s/4}^2 \cdot \frac{1+\log(p_*)}{p_*}2^{\tau/\lambda} \leq 8s2^{\tau/\lambda}\frac{\binom{s/2}{s/4}^2}{p}.$$

Conditioning on such a p, at least a $3/4$ fraction of $T \in \binom{S}{s/2}$ satisfies $C_p(T) \leq 32s2^{\tau/\lambda}\binom{s/2}{s/4}^2/p$. Since $T \in \binom{S}{s/2}$ if and only if $S \setminus T \in \binom{S}{s/2}$, we get by the union bound that half of all $T \in \binom{S}{s/2}$ satisfy

$$C_p(T) \leq 32s2^{\tau/\lambda}\binom{s/2}{s/4}^2/p, \qquad C_p(S\setminus T) \leq 32s2^{\tau/\lambda}\binom{s/2}{s/4}^2/p. \tag{13}$$

We thus conclude that that assumption (ii) in Lemma 5 holds with probability at least $3/4$ over $p \in \mathscr{M}$.

Next let us consider assumption (i) in Lemma 5. We now take $\mathscr{M}$ to be the set of products pq of primes p in the interval $[p_*, 2p_*]$ and primes q in the interval $[q_*, 2q_*]$. By Lemma 2, we have $|\mathscr{M}| \geq \frac{p_*q_*}{\log(p_*)\log(q_*)} \geq p_*q_*/s^2$, so Lemma 7 implies

$$\mathbb{E}_{pq\in\mathscr{M}}[C_{pq}(S)] \leq \binom{s}{s/2}B + \binom{s}{s/2}^2\frac{s^2 2^{\tau/\lambda}}{p_*q_*} = \binom{s}{s/2}^2\left(\frac{1}{p_*q_*} + \frac{s^2 2^{\tau/\lambda}}{p_*q_*}\right).$$

Markov's inequality then implies that with probability at least $3/4$ over $pq \in \mathscr{M}$, we have

$$C_{pq}(S) \leq 64s^2 2^{\tau/\lambda}\frac{\binom{s}{s/2}^2}{pq}. \tag{14}$$

Taking the union bound, we conclude that with probability at least $1/2$ over the choice of p, q both assumptions (i) and (ii) in Lemma 5 hold, with $\gamma = 64s^2 2^{\tau/\lambda}$. ◀

4.3 Combining the Two Parts

By Lemma 8, we have for a random independent pair p, q that S is γ-well-spread relative to the modulus pq, and for at least half of all $T \in \binom{S}{s/2}$, it holds that both T and $S \setminus T$ are γ-well-spread relative to the modulus p, with $\gamma = 2^{6+\tau/\lambda}s^2$. Then Lemma 5 gives that $|h_{p,q}(\mathscr{D})| \geq p^3q/(2(2^{6+\tau/\lambda}s^2)^3) = 2^{-(19+3\tau/\lambda)}s^{-6}p^3q$.

5 The Search Subroutine

Our goal in this section is to build the search subroutine in Lemma 4. We build the subroutine from the bottom up, starting in §5.1 from subroutines that build bottom-level candidate partial solutions of size $s/4$, then in §5.2 proceeding to the mid-level subroutines that assemble the candidate partial solutions of size $s/2$, and finally arrive in §5.3 at root-level node that assembles all the sets S of size s required by Lemma 4.

5.1 Subroutine for Bottom-Level Nodes

Let $p \in \mathbb{Z}_{\geq 1}$ be fixed. The following subroutine is executed in the four bottom-level nodes.

▶ **Lemma 9.** *There is a randomized algorithm for listing solutions $Z \in \binom{[n]}{s/4}$ to $a(Z) \equiv k \pmod p$ with the following properties:*

- **(i)** *For every fixed k, every solution Z gets listed by the algorithm with probability at least $1/\sqrt{n}$ (over the internal randomness of the algorithm).*
- **(ii)** *If k is picked uniformly at random, the expected running time of the algorithm over the choice of k is $O\big(n^2\binom{n/2}{s/8} + n^2\binom{n/2}{s/8}^2/p\big)$ and the expected number of solutions found is at most $\binom{n/2}{s/8}^2/p$.*

Proof. The algorithm starts by picking a random subset $\mathscr{N} \in \binom{[n]}{n/2}$.

Next, we construct $\mathscr{L} = \binom{\mathscr{N}}{s/8}$, $\mathscr{R} = \binom{[n]\setminus\mathscr{N}}{s/8}$, the subsets of $\mathscr{N}$ and $[n] \setminus \mathscr{N}$ of size $s/8$. For each $X \in \mathscr{L} \cup \mathscr{R}$ we compute the residue of $a(X)$ modulo p and sort the two lists in increasing order of residue.

Initialize $\mathscr{S}$ to an empty list. Using the sorted lists, for each $j \in \mathbb{Z}_p$ do the following. Iterate over all $X \in \mathscr{L}$ such that $a(X) \equiv j \pmod p$ and over all $Y \in \mathscr{R}$ such that $a(Y) \equiv k - j \pmod p$. (Note that we do this implicitly by simultaneously scanning the two lists, not with an explicit loop that considers each j in turn.) For each such pair we append $X \cup Y$ to $\mathscr{S}$.

Let us now analyze success probability and running time.

For success probability, note that any fixed solution $Z \in \binom{[n]}{s/4}$ gets split perfectly by $\mathscr{N}$ (i.e., $|\mathscr{N} \cap Z| = s/8$) with probability $\binom{n/2}{s/8}^2 / \binom{n}{s/4} \geq 1/\sqrt{n}$ over the choice of $\mathscr{N}$. Whenever this happens, the algorithm finds Z.

For the running time, constructing the lists $\mathscr{L}$ and $\mathscr{R}$ can be done with brute force in $O\big(n^2\binom{n/2}{s/8}\big)$ time and space.[8] The running time of the merge step is bounded by $O(n^2)$ times the number of solutions, which on average over a random $k \in \mathbb{Z}_p$ is $\binom{n/2}{s/8}^2/p$ (since the total number of solutions over all $k \in \mathbb{Z}_p$ is simply $|\mathscr{L}| \cdot |\mathscr{R}|$). ◀

5.2 Subroutine for Mid-Level Nodes

We now proceed to the subroutine executed by the two mid-level nodes. Let $M = pq$ for two distinct primes p, q.

▶ **Lemma 10.** *There is a randomized algorithm for listing solutions $(X, Y) \in \binom{[n]}{s/4}^2$ to $a(X) + a(Y) \equiv k_M \pmod M$ and $a(X) \equiv k_p \pmod p$ with $X \cap Y = \emptyset$, with the following properties:*

- **(i)** *For every fixed k_M, k_p, every solution (X, Y) gets listed by the algorithm with probability at least $1/n$ (over the internal randomness of the algorithm).*
- **(ii)** *If k_M, k_p are picked uniformly at random from $\mathbb{Z}_M$ and $\mathbb{Z}_p$, respectively, the expected running time of the algorithm over the choice of k_M, k_p is $O\big(n^2\binom{n/2}{s/8} + n^2\binom{n/2}{s/8}^2/p + n^2\binom{n/2}{s/8}^4/(Mp)\big)$ and the expected number of solutions found is at most $\binom{n/2}{s/8}^4/(Mp)$.*

Proof. The algorithm starts by picking $k_p \in \mathbb{Z}_p$ uniformly at random. It then executes the algorithm of Lemma 9 twice, with parameters $k = k_p$ and $k = (k_M - k_p) \bmod p$, yielding

[8] We are sorting $2^{\Omega(n)}$ items and comparing each pair of items takes $\Omega(n)$ time.

two lists $\mathscr{L}$ (of solutions $Z \in \binom{n}{s/8}$ to $a(Z) \equiv k_p \pmod p$) and $\mathscr{R}$ (of solutions $Z \in \binom{n}{s/8}$ to $a(Z) \equiv k_M - k_p \pmod p$).

We now merge these in a similar way as the proof of Lemma 9, except that we filter out all (X, Y) which are not disjoint and simply don't add them to the result.

The success probability follows because the two calls to the bottom-level algorithm are independent in terms of the internal randomness used by the calls.

For the running time, note that the two parameters k_p and $(k_M - k_p) \bmod p$ to the bottom-level algorithm are both uniformly random so that the expected sizes (over k_p and k_M) of $\mathscr{L}$ and $\mathscr{R}$ are at most $\binom{n/2}{s/8}^2/p$. Furthermore, since p divides M, $k_M \bmod p$ is uniformly random and the two parameters are independent, implying that the expectation of $|\mathscr{L}| \cdot |\mathscr{R}|$ is at most $\binom{n/2}{s/8}^4/p^2$.

By the Chinese Remainder Theorem, $k_M \bmod q$ is independent of $k_M \bmod p$, so conditioned on k_p, $k_M \bmod p$, $\mathscr{L}$ and $\mathscr{R}$, the expected running time of the merge step is $O(|\mathscr{L}| \cdot |\mathscr{R}|/q)$. Thus in overall expectation over the choice of k_p and k_M, the running time is $\binom{n/2}{s/8}^4/(p^2 q)$. ◀

5.3 The Root-Level Node

We are now ready to complete the proof of Lemma 4.

Proof of Lemma 4. Apply the algorithm of Lemma 10 twice, first with parameters $(k_M, k_p) = (k_\mathrm{T}, k_\mathrm{L})$, then with parameters $(k_M, k_p) = ((t - k_\mathrm{T}) \bmod M, k_\mathrm{R})$. Thus we get a list $\mathscr{L}$ of solutions $(X, Y) \in \binom{[n]}{s/4}^2$ such that $X \cap Y = \emptyset$, $a(X) + a(Y) \equiv k_M \pmod M$ and $a(X) = k_p \pmod p$. Similarly we have a list $\mathscr{R}$ of solutions $(Z, W) \in \binom{[n]}{s/4}^2$ such that $Z \cap W = \emptyset$, $a(Z) + a(W) \equiv t - k_M \pmod M$, $a(Z) \equiv k_\mathrm{R} \pmod p$.

We now merge these lists to construct solutions $(X, Y, Z, W) \in \binom{[n]}{s}$ to $a(X) + a(Y) + a(Z) + a(W) = t$, and if $(X \cup Y) \cap (Z \cup W) = \emptyset$, we output the solution $X \cup Y \cup Z \cup W$.

For every fixed choice of $(k_\mathrm{T}, k_\mathrm{L}, k_\mathrm{R})$ and solution $X \cup Y \cup Z \cup W$, the probability that we find the solution is at least $1/n^2$ by independence of the internal randomness employed by the two applications of Lemma 10.

By B-bounded concentration, for any choice $a(X) + a(Y)$ from $\mathscr{L}$, there are at most B solutions (Z, W) from $\mathscr{R}$ to $a(Z) + a(W) = t - a(X) - a(Y)$. This implies that the merge step runs in time $O(n^2(|\mathscr{L}| + |\mathscr{R}|)B)$. Since in expectation over the targets k. the sizes of $\mathscr{L}$ and $\mathscr{R}$ are bounded by $\binom{n/2}{s/8}^4/(Mp)$, it follows that the merge step runs in expected time $O\big(n^2\binom{n/2}{s/8}^4 B/(Mp)\big)$.

Substituting $s = \sigma n$ and and bounding the binomial coefficients from above with Stirling's formula [20], we obtain the desired running time bound (4) required by Lemma 4. This completes the proof of Lemma 4. ◀

Acknowledgements This research was funded in part by the European Research Council, Advanced Grant 226203 and Swedish Research Council, Grant 621-2012-4546 (P.A.), the European Research Council, Starting Grant 338077 "Theory and Practice of Advanced Search and Enumeration" (P.K.), the Academy of Finland, Grant 276864 "Supple Exponential Algorithms" (M.K.), and by the NWO VENI project 639.021.438 (J.N.).

References

1 Dimitris Achlioptas. Random satisfiability. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 245–270. IOS Press, 2009.

2 Per Austrin, Petteri Kaski, Mikko Koivisto, and Jussi Määttä. Space-time tradeoffs for Subset Sum: An improved worst case algorithm. In Fedor V. Fomin, Rusins Freivalds, Marta Z. Kwiatkowska, and David Peleg, editors, *ICALP (1)*, volume 7965 of *Lecture Notes in Computer Science*, pages 45–56. Springer, 2013.

3 Anja Becker. *The Representation Technique: Application to Hard Problems in Cryptography*. PhD thesis, Université de Versailles Saint-Quentin-en-Yvelines, 2012.

4 Anja Becker, Jean-Sébastien Coron, and Antoine Joux. Improved generic algorithms for hard knapsacks. In Kenneth G. Paterson, editor, *EUROCRYPT*, volume 6632 of *Lecture Notes in Computer Science*, pages 364–385. Springer, 2011.

5 Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, N. J., 1957.

6 Khodakhast Bibak. Additive combinatorics: With a view towards computer science and cryptography - an exposition. In Jonathan M. Borwein, Igor Shparlinski, and Wadim Zudilin, editors, *Number Theory and Related Fields*, pages 99–128. Springer, 2013.

7 Matthijs J. Coster, Antoine Joux, Brian A. Lamacchia, Andrew M. Odlyzko, Claus-Peter Schnorr, and Jacques Stern. Improved low-density subset sum algorithms. *Computational Complexity*, 2:111–128, 1992.

8 Itai Dinur, Orr Dunkelman, Nathan Keller, and Adi Shamir. Efficient dissection of composite problems, with applications to cryptanalysis, knapsacks, and combinatorial search problems. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 719–740. Springer, 2012.

9 Michael R. Fellows, Serge Gaspers, and Frances A. Rosamond. Parameterizing by the number of numbers. *Theory Comput. Syst.*, 50(4):675–693, 2012.

10 Abraham D. Flaxman and Bartosz Przydatek. Solving medium-density subset sum problems in expected polynomial time. In Volker Diekert and Bruno Durand, editors, *STACS 2005*, volume 3404 of *Lecture Notes in Computer Science*, pages 305–314. Springer Berlin Heidelberg, 2005.

11 Fedor V. Fomin and Dieter Kratsch. *Exact Exponential Algorithms*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2010.

12 Godfrey H. Hardy and Edward M. Wright. *An Introduction to the Theory of Numbers*. Oxford University Press, Oxford, sixth edition, 2008. Revised by D. R. Heath-Brown and J. H. Silverman, With a foreword by Andrew Wiles.

13 Ellis Horowitz and Sartaj Sahni. Computing partitions with applications to the knapsack problem. *J. Assoc. Comput. Mach.*, 21:277–292, 1974.

14 Nick Howgrave-Graham and Antoine Joux. New generic algorithms for hard knapsacks. In Henri Gilbert, editor, *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 235–256. Springer, 2010.

15 Russell Impagliazzo and Moni Naor. Efficient cryptographic schemes provably as secure as subset sum. *Journal of Cryptology*, 9(4):199–216, 1996.

16 Petteri Kaski, Mikko Koivisto, and Jesper Nederlof. Homomorphic hashing for sparse coefficient extraction. In Dimitrios M. Thilikos and Gerhard J. Woeginger, editors, *IPEC*, volume 7535 of *Lecture Notes in Computer Science*, pages 147–158. Springer, 2012.

17 Jeffrey C. Lagarias and Andrew M. Odlyzko. Solving low-density subset sum problems. *J. ACM*, 32(1):229–246, 1985.

18 Daniel Lokshtanov and Jesper Nederlof. Saving space by algebraization. In Leonard J. Schulman, editor, *STOC*, pages 321–330. ACM, 2010.

19 Phong Q. Nguyen, Igor E. Shparlinski, and Jacques Stern. Distribution of modular sums and the security of the server aided exponentiation. In Kwok-Yan Lam, Igor Shparlinski, Huaxlong Wang, and Chaoping Xing, editors, *Cryptography and Computational Number Theory*, volume 20 of *Progress in Computer Science and Applied Logic*, pages 331–342. Birkhäuser, 2001.

20 Herbert Robbins. A remark on Stirling's formula. *The American Mathematical Monthly*, 62(1):26–29, 1955.

21 Richard Schroeppel and Adi Shamir. A $T = O(2^{n/2})$, $S = O(2^{n/4})$ algorithm for certain NP-complete problems. *SIAM J. Comput.*, 10(3):456–464, 1981.

22 Terence Tao. The dichotomy between structure and randomness, arithmetic progressions, and the primes. In *International Congress of Mathematicians. Vol. I*, pages 581–608. Eur. Math. Soc., Zürich, 2007.

23 Terence Tao. Structure and randomness in combinatorics. In *FOCS*, pages 3–15. IEEE Computer Society, 2007.

24 Terence Tao. *Structure and Randomness*. American Mathematical Society, Providence, RI, 2008.

25 Terence Tao and Van Vu. *Additive Combinatorics*, volume 105 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge, 2006.

26 Terence Tao and Van Vu. A sharp inverse Littlewood-Offord theorem. *Random Structures Algorithms*, 37(4):525–539, 2010.

27 Luca Trevisan. Pseudorandomness in computer science and in additive combinatorics. In *An irregular mind*, volume 21 of *Bolyai Soc. Math. Stud.*, pages 619–650. János Bolyai Math. Soc., Budapest, 2010.

28 Gerhard J. Woeginger. Exact algorithms for NP-hard problems: A survey. In Michael Jünger, Gerhard Reinelt, and Giovanni Rinaldi, editors, *Combinatorial Optimization*, volume 2570 of *Lecture Notes in Computer Science*, pages 185–208. Springer, 2001.