Austrin, Per; Kaski, Petteri; Koivisto, Mikko; Nederlof, Jesper

Dense Subset Sum may be the hardest

Published in:
Leibniz International Proceedings in Informatics

DOI:
10.4230/LIPIcs.STACS.2016.13

Published: 01/02/2016

Document Version
Publisher's PDF, also known as Version of record

Published under the following license:
CC BY

Please cite the original version:
Austrin, P., Kaski, P., Koivisto, M., & Nederlof, J. (2016). Dense Subset Sum may be the hardest. In N. Ollinger, & H. Vollmer (Eds.), *Leibniz International Proceedings in Informatics: LIPIcs* (Vol. 47, pp. 1-12). [13] Schloss Dagstuhl- Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing. https://doi.org/10.4230/LIPIcs.STACS.2016.13

Dense Subset Sum May Be the Hardest

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**
j.nederlof@tue.nl

Abstract

The Subset Sum problem asks whether a given set of n positive integers contains a subset of elements that sum up to a given target t. It is an outstanding open question whether the $O^*(2^{n/2})$-time algorithm for Subset Sum by Horowitz and Sahni [J. ACM 1974] can be beaten in the worst-case setting by a "truly faster", $O^*(2^{(0.5-\delta)n})$-time algorithm, with some constant $\delta > 0$. Continuing an earlier work [STACS 2015], we study Subset Sum parameterized by the maximum bin size β, defined as the largest number of subsets of the n input integers that yield the same sum. For every $\epsilon > 0$ we give a truly faster algorithm for instances with $\beta \leq 2^{(0.5-\epsilon)n}$, as well as instances with $\beta \geq 2^{0.661n}$. Consequently, we also obtain a characterization in terms of the popular density parameter $n/\log_2 t$: if all instances of density at least 1.003 admit a truly faster algorithm, then so does every instance. This goes against the current intuition that instances of density 1 are the hardest, and therefore is a step toward answering the open question in the affirmative. Our results stem from a novel combinatorial analysis of mixings of earlier algorithms for Subset Sum and a study of an extremal question in additive combinatorics connected to the problem of Uniquely Decodable Code Pairs in information theory.

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.2016.13

1 Introduction

The Subset Sum problem and its generalization to the Knapsack problem are two of the most famous NP-complete problems. In the Subset Sum problem, we are given positive integers $w_1, w_2, \ldots, w_n, t \in \mathbb{Z}$ as input, and need to decide whether there exists a subset $X \subseteq [n]$ with $\sum_{j \in X} w_j = t$. In the Knapsack problem, we are additionally given integers $v_1, v_2, \ldots, v_n$ and are asked to find a subset $X \subseteq [n]$ maximizing $\sum_{j \in X} v_j$ subject to the constraint $\sum_{j \in X} w_j \leq t$. While the study of Subset Sum is, among others, motivated by cryptographic applications or balancing problems, Knapsack has numerous applications in combinatorial optimization. We study the exact worst-case time complexity of these problems. The earliest and probably most important algorithms for both problems are simple applications of dynamic programming, pioneered by Bellman [5], solving both problems in $O^*(t)$ time (where the $O^*(\cdot)$ notation suppresses factors polynomial in the input size). In terms of n, the best algorithms for both problems are due to Schroeppel and Shamir [18], using $O^*(2^{n/2})$ time and $O^*(2^{n/4})$ space, based on the *meet-in-the-middle* technique by Horowitz and Sahni [9]. Nederlof et al. [16] show that there is an $O^*(T^n)$-time, $O^*(S^n)$-space algorithm for Subset Sum if and only if there is an $O^*(T^n)$-time, $O^*(S^n)$-space algorithm for Knapsack. A major open question since the paper by Horowitz and Sahni [9] is whether we can do "truly faster" for both problems:

Open Question 1: Can Subset Sum be solved in $O^*\big(2^{(0.5-\delta)n}\big)$ time for some constant $\delta > 0$?

In this paper we discuss Monte Carlo algorithms in the following sense: the algorithm never returns false positives and constructs solutions of yes-instances with at least inverse polynomial probability. All randomized algorithms discussed in this paper are of this type, but for Open Question 1 we would be satisfied with two-sided error as well.

Zooming out, one motivation of this question is as follows. It is commonly believed that there are no polynomial time or even sub-exponential time algorithms for Subset Sum. So how fast can the fastest algorithm be? It would be an elegant situation if the simple meet-in-the-middle algorithm was optimal. But this would also be quite surprising, and so we aim to show that at least this is not the case.

In 2010, Howgrave-Graham and Joux [10] gave an algorithm that answered Open Question 1 in the affirmative in an *average case* setting. To state their result, let us describe the setting where it applies. The *density* of a Subset Sum instance is defined as $n/\log_2 t$. A random instance of density $d > 0$ is constructed by fixing $t \approx 2^{n/d}$ and picking the integers $w_1, \dots, w_n, t$ independently and uniformly at random between 1 and $2^{n/d}$. Howgrave-Graham and Joux [10] showed that random instances of density 1 can be solved in $O^*(2^{0.311n})$ time, and later this has been improved to $O^*(2^{0.291n})$ time by Becker et al. [4]. These results resolve Open Question 1 in the average case setting since Impagliazzo and Naor [11] showed that random instances are the *hardest when they have density* 1. Indeed, a vast body of research has given better algorithms for random instances with density deviating from 1, like reductions of sparse instances to the shortest vector problem (e.g. [14, 6]) and the algorithm by Flaxman and Przydatek [7].

The algorithms discussed thus far all use exponential space, which can be a serious bottleneck. Therefore many studies also emphasize the setting where the algorithm is restricted to using polynomial space. It is known that the running time of the dynamic programming based algorithms can be achieved also in polynomial space: Lokshtanov and Nederlof [15] give polynomial space algorithms solving Subset Sum in $O^*(t)$ time and Knapsack in pseudo-polynomial time. On the other hand, in terms of n, no polynomial space algorithm significantly faster than naïvely going through all 2^n subsets is known, and the following has been stated as an open problem by a number of researchers (see e.g. [20, 8]):

Open Question 2: Can Subset Sum be solved in polynomial space and $O^*\big(2^{(1-\delta)n}\big)$ time for some constant $\delta > 0$?

1.1 Our results

We aim to make progress on Open Question 1, and show that a large class of instances can be solved truly faster. An optimist may interpret this as an indication that truly faster algorithms indeed exist, while a pessimist may conclude the remaining instances must be the (strictly) hardest instances.

Algorithmic Results. To define classes of instances that admit truly faster algorithms, we consider several natural parameters. The key parameter that seems to capture the range of our algorithmic technique the best is the *maximum bin size* $\beta(w) = \max_{x\in\mathbb{Z}} |\{S \subseteq [n] : \sum_{i\in S} w_i = x\}|$. Our main technical result is:

▶ **Theorem 1.1.** *There exists a Monte Carlo algorithm that, for any* $0 \le \epsilon \le 1/6$*, solves all instances of* Subset Sum *with* $\beta(w) \le 2^{(0.5-\epsilon)n}$ *in* $O^*\big(2^{(0.5-\epsilon/4+3\epsilon^2/4)n}\big)$ *time.*

We have not optimized the precise constants in Theorem 1.1 – the main message is that any instance with bin size up to $2^{(0.5-\epsilon)n}$ can be solved in time $2^{(0.5-\Omega(\epsilon))n}$. For $\epsilon \ge 1/6$, the running time of $2^{23n/48}$ obtained for $\epsilon = 1/6$ is still valid since $2^{(0.5-1/6)n}$ remains an upper bound on $\beta(w)$. In a previous work [2], we solved Subset Sum in time $O^*(2^{0.3399n}\beta(w)^4)$, which is faster than Theorem 1.1 for small $\beta(w)$, but Theorem 1.1 shows that we can beat the meet-in-the-middle bound for a much wider class of instances.

From the other end, we also prove that when the maximum bin size becomes too large, we can again solve Subset Sum truly faster:

▶ **Theorem 1.2.** *There exist a constant* $\delta > 0$ *and a deterministic algorithm that solves all instances of* Subset Sum *with* $\beta(w) \ge 2^{0.661n}$ *in* $O^*\big(2^{(0.5-\delta)n}\big)$ *time.*

Combinatorial Results. Given Theorem 1.1, the natural question is how instances with $\beta(w) \ge 2^{0.5n}$ look like. This question is an instantiation of the inverse Littlewood-Offord problem, a subject well-studied in the field of additive combinatorics. Ideally we would like to find structural properties of instances with $\beta(w) \ge 2^{0.5n}$, that can be algorithmically exploited by other means than Theorem 1.1 in order to resolve Open Question 1 in the affirmative. While there is a large amount of literature on the inverse Littlewood-Offord problem, the typical range of $\beta(w)$ studied there is $\beta(w) = 2^n/\operatorname{poly}(n)$ which is not relevant for our purposes. However, we did manage to determine additional properties that any instance that is not solved by Theorem 1.1 must satisfy.

In particular, we study a different natural parameter, the *number of distinct sums* generated by w, defined as $|w(2^{[n]})| = \{w(X) : X \subseteq [n]\}$ (where we denote $w(X) = \sum_{i\in X} w_i$). This parameter can be viewed as a measure of the "true" density of an instance, in the following sense. An instance with density $d = n/\log_2 t$ has $|w(2^{[n]})| \le n2^{n/d}$ (assuming without loss of generality that $t \le \max_i w_i$). On the other hand, by standard hashing arguments (e.g., Lemma 2.2 with $B = 10|w(2^{[n]})|$), any instance can be hashed down to an equivalent instance of density roughly $n/\log_2 |w(2^{[n]})|$.

The relationship between $|w(2^{[n]})|$ and $\beta(w)$ is more complicated. Intuitively, one would expect that if one has so much concentration that $\beta(w) \ge 2^{0.5n}$, then w should not generate too many sums. We are not aware of any such results from the additive combinatorics literature. However, by establishing a new connection to *Uniquely Decodable Code Pairs*, a well-studied object in information theory, we can derive the following bound.

▶ **Lemma 1.3.** *If* $|w(2^{[n]})| \ge 2^{0.997n}$ *then* $\beta(w) \le 2^{0.4996n}$.

Unfortunately, we currently do not know how to algorithmically exploit $|w(2^{[n]})| \leq 2^{0.997n}$. But we do know how to exploit a set S with $|S| \leq n/2$ and $|w(2^S)| \leq 2^{0.4999n}$ (see Lemma 3.2). This suggests the question of how large $\beta(w)$ can be in instances lacking such an S, and we prove the following bound.

▶ **Lemma 1.4.** *There is a universal constant $\delta > 0$ such that the following holds for all sufficiently large n. Let S, T be a partition of $[n]$ with $|S| = |T| = n/2$ such that $|w(2^S)|, |w(2^T)| \geq 2^{(1/2-\delta)n}$. Then $\beta(w) \leq 2^{0.661n}$.*

Further Consequences. Combining Lemma 1.3 and Theorem 1.1, we see directly that instances that generate almost 2^n distinct sums can be solved faster than $2^{0.5n}$.

▶ **Theorem 1.5.** *There exists a Monte Carlo algorithm that solves all instances of* Subset Sum *with $|w(2^{[n]})| \geq 2^{0.997n}$ in time $O^*(2^{0.49991n})$.*

Combining this with the view described above of $|w(2^{[n]})|$ as a refined version of the density of an instance, we have the following result, to support the title of our paper:

▶ **Theorem 1.6.** *Suppose there exist a constant $\epsilon > 0$ and an algorithm that solves all* Subset Sum *instances of density at least* 1.003 *in time $O^*(2^{(0.5-\epsilon)n})$. Then there exists a Monte Carlo algorithm that solves* Subset Sum *in time $O^*\big(2^{\max\{0.49991, 0.5-\epsilon\}n}\big)$.*

After the result by Howgrave-Graham and Joux [10], this may be a next step towards resolving Open Question 1. Intuitively, one should be able to exploit the fact that the integers in a dense instance have fewer than n bits. For example, even if only the target is picked uniformly at random, in expectation there will be an exponential number of solutions, which can easily be exploited.[1]

Finally, let us note a somewhat curious consequence of our results. As mentioned earlier, in the context of Open Question 2, it is known that the $O^*(2^{n/d})$ running time for instances of density d achieved through dynamic programming can be achieved in polynomial space [15] (see also [13, Theorem 1(a)]). Combining this with Corollary 1.5 and hashing, we directly get the following "interleaving" of Open Questions 1 and 2.

▶ **Corollary 1.7.** *There exist two Monte Carlo algorithms, one running in $O^*(2^{0.49991n})$ time and the other in $O^*(2^{0.999n})$ time and polynomial space, such that every instance of* Subset Sum *is solved by at least one of the algorithms.*

Organization of the paper. This paper is organized as follows: In Section 2 we review some preliminaries. In Section 3, we provide the proofs of our main algorithmic results. In Section 4 we prove two combinatorial lemmas. In Section 5 we give the proof for Theorem 1.6. Finally we end with some discussion on in Section 6.

2 Preliminaries

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 $x - y$. Throughout this paper, $w_1, w_2, \ldots, w_n, t$ will denote the input

[1] For example, assuming there are at least $2^{\sigma n}$ solutions for a constant $\sigma \geq 0$, use a dynamic programming table data structure to randomly sample the subsets in the congruence class t mod q for q a random prime with about $(1-\sigma)n/2$ bits within linear time per sample. A solution is found within $O^*(2^{(1-\sigma)n/2})$ samples with high probability.

integers of a Subset Sum instance. We associate the set function $w : 2^{[n]} \to \mathbb{Z}$ with these integers by letting $w(X) = \sum_{i \in X} w_i$, and for a set family $\mathcal{F} \subseteq 2^{[n]}$ we write $w(\mathcal{F})$ for the image $\{w(X) : X \in \mathcal{F}\}$.

For $0 \leq x_1, x_2, \ldots, x_\ell \leq 1$ with $\sum_{i=1}^{\ell} x_i = 1$ we write $h(x_1, x_2, \ldots, x_\ell) = \sum_{i=1}^{\ell} -x_i \log_2 x_i$ for the entropy function. Here, $0 \log_2 0$ should be interpreted as 0. We shorthand $h(x, 1-x)$ with $h(x)$. We routinely use the standard fact (easily proved using Stirling's formula) that for non-negative integers $n_1, \ldots, n_\ell$ (where ℓ is a constant) summing to n, it holds that $\binom{n}{n_1, \ldots, n_\ell} = 2^{h(n_1/n, \ldots, n_\ell/n)n} \cdot \mathrm{poly}(n)$.

▶ **Claim 2.1.** *For every sufficiently large integer r the following holds. If p is a prime between r and $2r$ selected uniformly at random and x is a nonzero integer, then p divides x with probability at most $(\log_2 x)/r$.*

▶ **Lemma 2.2** (Bit-length reduction)**.** *There exists a randomized algorithm that takes as input a Subset Sum instance $w_1, w_2, \ldots, w_n, t \in \mathbb{Z}$ and an integer $B \in \mathbb{Z}$, and in time $O^*(1)$ outputs a new Subset Sum instance $w'_1, w'_2, \ldots, w'_n, t' \in \mathbb{Z}$ such that with probability at least inversely polynomial in the input size, the following properties all simultaneously hold.*

1. $0 \leq w'_1, w'_2, \ldots, w'_n, t' < 4nB \log_2 B$.
2. *If $B \geq 10 \cdot |w(2^{[n]})|$, then $X \subseteq [n]$ satisfies $w(X) = t$ if and only if $w'(X) = t'$.*
3. *If $B \geq 10 \cdot |w(2^{[n]})|$, then $|w(2^{[n]})|/2 \leq |w'(2^{[n]})| \leq n|w(2^{[n]})|$.*
4. *If $B \geq 5 \cdot |w(2^{[n]})|^2$, then $\beta(w)/n \leq \beta(w') \leq \beta(w)$.*

The proofs of Claim 2.1 and Lemma 2.2 use standard techniques and are presented in the full version [3].

3 Algorithmic Results

This section establishes Theorems 1.1 and 1.2. We begin with two lemmas showing how one can exploit a subset of the input integers if it generates either many or few distinct sums. The case of many sums is the main technical challenge and addressed by the following result, which is our main algorithmic contribution.

▶ **Lemma 3.1.** *There is a randomized algorithm that, given positive integers $w_1, \ldots, w_n, t \leq 2^{O(n)}$ and a set $M \in \binom{[n]}{\mu n}$ satisfying $\mu \leq 0.5$ and $|w(2^M)| \geq 2^{\gamma |M|}$ for some $\gamma \in [0, 1]$, finds a subset $X \subseteq [n]$ satisfying $w(X) = t$ with probability at least inversely polynomial in the input size (if such an X exists) in time $O^*\big(2^{(0.5+0.8113\mu-\gamma\mu)n} + \beta(w)2^{(1.5-\gamma)\mu n}\big)$.*

The proof is given in Section 3.1. Informally, it uses an algorithm that simultaneously applies the meet-in-the-middle technique of Horowitz and Sahni [9] on the set $[n] \setminus M$ and the "representation technique" of Howgrave-Graham and Joux [10] on the set M. Specifically, we pick an arbitrary equi-sized partition L, R of $[n] \setminus M$ and construct lists $\mathcal{L} \subseteq 2^{L \cup M}$ and $\mathcal{R} \subseteq 2^{R \cup M}$. Note that without restrictions on $\mathcal{L}$ and $\mathcal{R}$, one solution X is witnessed by $2^{|M \cap X|}$ pairs (S, T) from $\mathcal{L} \times \mathcal{R}$ in the sense that $S \cup T = X$. Now the crux is that since M generates many sums, $M \cap X$ generates many sums (say $2^{\pi |M|}$): this allows us to uniformly choose a congruence class t_L of $\mathbb{Z}_p$ where p is a random prime of order $2^{\pi |M|}$ and restrict attention only to sets $S \subseteq L \cup M$ and $T \subseteq R \cup M$ such that $w(S) \equiv_p t_L$ and $w(T) \equiv_p t - t_L$, while still finding solutions with good probability. This ensures that the to-be-constructed lists $\mathcal{L}$ and $\mathcal{R}$ are small enough. As an indication for this, note that if $|M \cap X| = |M|/2$ and $|w(\binom{M \cap X}{|M|/4})|$ is $\Omega(2^{|M|/2})$, the expected sizes of $\mathcal{L}$ and $\mathcal{R}$ are at most $2^{((1-\mu)/2+h(1/4)\mu-\mu/2)n} \leq 2^{(1/2-0.18\mu)n}$.

In contrast to Lemma 3.1, it is straightforward to exploit a small subset that generates few sums:

▶ **Lemma 3.2.** *There is a deterministic algorithm that, given positive integers $w_1, \ldots, w_n, t$ and a set $M \in \binom{[n]}{\mu n}$ satisfying $\mu \leq 0.5$ and $|w(2^M)| \leq 2^{\gamma|M|}$ for some $\gamma \in [0,1]$, finds a subset $X \subseteq [n]$ satisfying $w(X) = t$ (if such an X exists) in time $O^*\big(2^{\frac{1-\mu(1-\gamma)}{2}n}\big)$.*

Proof. Let L be an arbitrary subset of $[n] \setminus M$ of size $\frac{1-\mu(1-\gamma)}{2}n$ and let $R = [n] \setminus L$. Then $|w(2^L)| \leq 2^{|L|} = 2^{\frac{1-\mu(1-\gamma)}{2}n}$, and

$$|w(2^R)| \leq |w(2^M)| \cdot |w(2^{[n]\setminus L \setminus M})| \leq 2^{\gamma\mu n} 2^{\left(1-\frac{1-\mu(1-\gamma)}{2}-\mu\right)n} = 2^{\frac{1-\mu(1-\gamma)}{2}n}.$$

Now apply routine dynamic programming to construct $w(2^L)$ in time $O^*(|w(2^L)|)$ and $w(2^R)$ in time $O^*(|w(2^R)|)$; build a look-up table data structure for $w(2^L)$, and for each $x \in w(2^R)$, check in $O(n)$ time whether $t - x \in w(2^L)$. ◀

Given these lemmas, we are now in the position to exploit small bins:

Proof of Theorem 1.1. We start by preprocessing the input with Lemma 2.2, taking $B = 2^{3n} \gg |w(2^{[n]})|^2$. Let $\gamma = 1 - \epsilon/2, \mu = 3\epsilon/2$, and partition $[n]$ into $1/\mu$ parts $M_1, \ldots, M_{1/\mu}$ of size at most μn arbitrarily. We distinguish two cases. First, suppose that $|w(2^{M_i})| \geq 2^{\gamma\mu n}$ for some M_i (note that this can be easily determined within the claimed time bound). We then apply the algorithm of Lemma 3.1 with $M = M_i$ and solve the instance (with probability $\Omega^*(1)$) in time

$$O^*\Big(2^{(0.5+0.8113\mu-\gamma\mu)n} + \beta(w)2^{(1.5-\gamma)\mu n}\Big).$$

The coefficient of the exponent of the first term is $0.5 + 0.8113 \cdot 3\epsilon/2 - (1-\epsilon/2) \cdot 3\epsilon/2 = 0.5 - 0.28305\epsilon + 0.75\epsilon^2$. The coefficient of the exponent of the second term is $0.5 - \epsilon + (1.5 - (1-\epsilon/2)) \cdot 3\epsilon/2 = 0.5 - \epsilon/4 + 0.75\epsilon^2$.

Second, suppose that $|w(2^{M_i})| \leq 2^{\gamma\mu n}$ for all i. Let $L = \bigcup_{i=1}^{\frac{1}{2\mu}} M_i$ and $R = [n] \setminus M$. We see that

$$|w(2^L)| \leq \prod_{i \leq \frac{1}{2\mu}} |w(2^{M_i})| \leq 2^{\gamma n/2} \qquad \text{and} \qquad |w(2^R)| \leq \prod_{i > \frac{1}{2\mu}} |w(2^{M_i})| \leq 2^{\gamma n/2}.$$

Using standard dynamic programming to construct $w(2^L)$ and $w(2^R)$ in $O^*(|w(2^L)|)$ and $O^*(|w(2^R)|)$ time, we can therefore solve the instance within $O^*(2^{\gamma n/2}) = O^*(2^{(0.5-\epsilon/4)n})$ time using linear search. ◀

Exploiting large bins is easy using Lemma 1.4 (the proof of Lemma 1.4 is given in Section 4):

Proof of Theorem 1.2. Pick an arbitrary equi-sized partition S, T of $[n]$. By the contrapositive of Lemma 1.4, one of S and T generates at most $2^{(1/2-\delta)n}$ sums. Applying Lemma 3.2 with the set in question as M, we get a running time of $O^*\big(2^{(1-\delta)n/2}\big)$. ◀

3.1 Proof of Lemma 3.1

We now prove Lemma 3.1. Let $s := |X \cap M|$. Without loss of generality, we may assume that $s \geq |M|/2$ (by considering the actual target t and the complementary target $t' := w([n]) - t$). We may further assume that s is known by trying all $O(n)$ possible values. The algorithm is listed in Algorithm 1.

```
Algorithm A$(w_1, \dots, w_n, t, M, s, \gamma)$                    Assumes $|w(2^M)| \geq 2^{\gamma|M|}$
Output: yes, if there exists an $X \subseteq [n]$ with $w(X) = t$ and $|X \cap M| = s$
 1: Let $\sigma = s/|M|$
 2: Let $\pi = \gamma - 1 + \sigma$
 3: Pick a random prime $p$ satisfying $2^{\pi|M|} \leq p \leq 2^{\pi|M|+1}$
 4: Pick a random number $0 \leq t_L \leq p-1$
 5: for all $0 \leq s_1 \leq s_2 \leq |M|$ such that $s_1 + s_2 = s$ do
 6:     Let $\sigma_1 = s_1/|M|, \sigma_2 = s_2/|M|$
 7:     Let $\lambda = (1-\mu)/2 + \big(h(\sigma/2) - h(\sigma_1)\big)\mu$
 8:     Let $L, R$ be an arbitrary partition of $[n] \setminus M$ such that $|L| = \lceil \lambda n \rceil$
 9:     Construct $\mathcal{L} = \{S \in 2^{L \cup M} : w(S) \equiv_p t_L$ and $|S \cap M| = s_1\}$
10:     Construct $\mathcal{R} = \{T \in 2^{R \cup M} : w(T) \equiv_p t - t_L$ and $|T \cap M| = s_2\}$
11:     for all $(S,T) \in \mathcal{L} \times \mathcal{R}$ such that $w(S) + w(T) = t$ do
12:         if $S \cap T = \emptyset$ then return yes
13: return no
```

Algorithm 1 Exploiting a small subset generating many sums.

Expected Running time. We will analyze the expected running time of Algorithm 1 in two parts: (i) the generation of the lists $\mathcal{L}$ and $\mathcal{R}$ on Lines 9 and 10, and (ii) the iteration over pairs in $\mathcal{L} \times \mathcal{R}$ in Line 11 (the typical bottleneck). Let $W_L := 2^{|L|}\binom{M}{s_1} \leq 2^{\lambda n} 2^{h(\sigma_1)\mu n} = 2^{((1-\mu)/2 + h(\sigma/2)\mu)n}$ denote the size of the search space for $\mathcal{L}$.

▶ **Proposition 3.3.** *The lists $\mathcal{L}$ and $\mathcal{R}$ in Lines 9 and 10 can be constructed in expected time $O^*\big(W_L^{1/2} + W_L/2^{\pi\mu n}\big)$, where the expectation is over the choice of p and t_L.*

Proof. By splitting the search space for $\mathcal{L}$ appropriately, we get two "halves" each of which has size $W_L^{1/2}$. Specifically, we arbitrarily pick a subset $L_1 \subseteq L$ of size $\lambda_1 n$ with $\lambda_1 = (\lambda + h(\sigma/2)\mu)/2$ and generate using brute-force $w(2^{L_1})$ and $w(\mathcal{L}_2)$ where $\mathcal{L}_2 = \{Y \cup Z : Y \subseteq L \setminus L_1$ and $Z \in \binom{M}{s_1}\}$. Then we store $w(2^{L_1})$ in a dictionary data structure and, for each sum $x \in w(\mathcal{L}_2)$, we look up all solutions with sum $t - x \bmod p$ in the dictionary of $w(2^{L_1})$ and list for such a pair its union. This yields a running time of $O^*(|\mathcal{L}| + W_L^{1/2})$. The expected size of $|\mathcal{L}|$ over the random choices of t_L is $\mathbb{E}[|\mathcal{L}|] \leq O(W_L/2^{\pi\mu n})$.

The analysis for $\mathcal{R}$ is analogous and we get a running time of $O^*\big(W_R^{1/2} + W_R/2^{\pi\mu n}\big)$ where $W_R := 2^{|R|}\binom{|M|}{s_2}$. Let $\rho = |R|/n$. Since $h(\cdot)$ is concave and, in particular, $h(\sigma_1) + h(\sigma_2) \leq 2h(\sigma/2)$, we then have (up to a negligible term caused by rounding λn to an integer)

$$\rho = 1 - \mu - \lambda = (1-\mu)/2 - \big(h(\sigma/2) - h(\sigma_1)\big)\mu \leq (1-\mu)/2 + (h(\sigma/2) - h(\sigma_2))\mu\,.$$

Thus the case of R is symmetric to the situation for L and $W_R \leq 2^{((1-\mu)/2 + h(\sigma/2)\mu)n} = O^*(W_L)$. ◀

The term $W_L/2^{\pi\mu n}$ can be bounded by using the definition of $\pi = \gamma - 1 + \sigma$ and we get $W_L/2^{\pi\mu n} = 2^{(\frac{1}{2} + \mu(\frac{1}{2} + h(\sigma/2) - \gamma - \sigma))n}$. Since $1/2 + h(\sigma/2) - \sigma$ subject to $1/2 \leq \sigma \leq 1$ is maximized at $\sigma = 1/2$ where it is $h(1/4) \leq 0.8113$, we have that $W_L/2^{\pi\mu n} \leq 2^{(0.5 + 0.8113\mu - \gamma\mu)n}$.

The term $W_L^{1/2}$ is naively bounded by $2^{(1+\mu)n/4}$, which is dominated by the term $O^*\big(2^{(0.5+0.8113\mu-\gamma\mu)n}\big)$ since $\mu \leq 1/2$ and $\gamma \leq 1$. It follows that Line 9 and Line 10 indeed run within the claimed time bounds.

▶ **Proposition 3.4.** *The expected number of pairs considered in Line 11 is $O^*\big(\beta(w)2^{\mu(1.5-\gamma)n}\big)$, where the expectation is over the choice of p and t_L.*

Proof. Define $\mathcal{B} = \big\{(P,Q) \in 2^{[n]} \times 2^M : w(P) + w(Q) = t\big\}$, and note that the set of pairs $(S,T) \in 2^{L \cup M} \times 2^{R \cup M}$ satisfying $w(S) + w(T) = t$ are in one-to-one correspondence with pairs in $\mathcal{B}$ (by the map $(S,T) \mapsto (S \cup (T \cap R), T \cap M)$). Furthermore, the size of $\mathcal{B}$ is bounded by $|\mathcal{B}| \leq \beta(w)2^{|M|}$: for each of the $2^{|M|}$ possible choices of Q, there are at most $\beta(w)$ subsets R that sum to $t - w(Q)$.

Any given pair $(S,T) \in 2^{L\cup M} \times 2^{R \cup M}$ satisfying $w(L) + w(R) = t$ is considered only if $w(S) \equiv_p t_L$, which happens with probability $O(2^{-\pi n})$ (over the uniformly random choice of t_L). Thus the expected number of pairs considered in Line 11 is upper bounded by $O(|B|/2^{\pi\mu n}) = O(\beta(w)2^{\mu(1-\pi)n}) = O(\beta(w)2^{\mu(2-\sigma-\gamma)n})$. Using $\sigma \geq 1/2$, the desired bound follows. ◀

Succes Probability. To establish Lemma 3.1, we run Algorithm 1 for $\Omega(|M|)$ times the expected number of computation steps and return **no** if it did not terminate yet. By our previous analysis it is clear that this algorithm runs within the required time bound, and clearly it only return **yes** if a solution is found on Line 12.

Now suppose there exists an $X \subseteq [n]$ with $w(X) = t$ and $|X \cap M| = s$. It remains to lower bound the probability that the modified algorithm return **yes**. Note that by Markov's inequality we return **no** due to premature termination with probability at most $O(1/|M|)$, so by a union bound it will be sufficient to lower bound the probability that Algorithm 1 returns **yes** by $\Omega(1/|M|)$.

To this end, note that $2^{\gamma|M|} \leq |w(2^M)| \leq |w(2^{M\cap X})| \cdot |w(2^{M\setminus X})|$, and since $|w(2^{M\setminus X})| \leq 2^{|M|-s}$, we have that $|w(2^{M\cap X})| \geq 2^{\gamma|M|-(1-\sigma)|M|} = 2^{\pi|M|}$.

Thus there must exist positive $s_1 + s_2 = s$ such that $|w(\binom{M\cap X}{s_1})| \geq 2^{\pi|M|}/|M|$. Let us focus on the corresponding iteration of Algorithm 1. Let $w_L := w(X \cap L)$ be the contribution of L to the solution X. We claim that in this iteration, the following holds.

▶ **Proposition 3.5.**

$$\Pr\left[\exists Q \in \binom{M \cap X}{s_1} : w(Q) \equiv_p t_L - w_L\right] \geq \Omega\left(\frac{1}{|M|}\right). \tag{1}$$

Note that, conditioned on the event $\exists Q \in \binom{M\cap X}{s_1} : w(Q) \equiv_p t_L - w_L$, Algorithm 1 will include $S := Q \cup (L \cap X)$ in $\mathcal{L}$ and $T := X \setminus S$ in $\mathcal{R}$ and recover X. Therefore, this concludes the proof of Lemma 3.1.

Proof. Let $\mathcal{F} \subseteq \binom{M\cap X}{s_1}$ be a maximal injective subset, i.e., satisfying $|\mathcal{F}| = |w(\mathcal{F})| = |w\left(\binom{M\cap X}{s_1}\right)| \geq \Omega^*(2^{\pi|M|})$. Let $c_i = |\{Y \in \mathcal{F} : w(Y) \equiv_p i\}|$ be the number of sets from $\mathcal{F}$ in the i'th bin mod p. Our goal is to lower bound the probability that $c_{t_L - w_L} > 0$ (where $t_L - w_L$ is taken modulo p). We can bound the expected ℓ^2 norm (e.g., the number of collisions) by

$$\mathbb{E}\left[\sum_i c_i^2\right] = \sum_{Y,Z\in\mathcal{F}} \Pr\big[p \text{ divides } w(Y) - w(Z)\big] \leq |\mathcal{F}| + O^*\big(|\mathcal{F}|^2/2^{\pi|M|}\big), \tag{2}$$

where the inequality uses Claim 2.1 and the assumption that the w_i's are $2^{O(n)}$. By Markov's inequality, $\sum_i c_i^2 \leq O^*(|\mathcal{F}|^2/2^{\pi|M|})$ with probability at least $\Omega^*(1)$ over the choice of p (here we used $|\mathcal{F}| = \Omega^*(2^{\pi|M|})$ to conclude that the second term in (2) dominates the first). Conditioned on this, Cauchy-Schwarz implies that the number of non-zero c_i's is at least $|\mathcal{F}|^2/\sum_i c_i^2 \geq \Omega^*(2^{\pi|M|})$. When this happens, the probability that $c_{t_L - w_L} > 0$ (over the uniformly random choice of t_L) is $\Omega^*(1)$. ◀

4 Combinatorial Results (Lemma 1.3 and Lemma 1.4)

In this section we provide two non-trivial quantitative relations between several structural parameters of the weights. Our results are by no means tight, but will be sufficient for proving our main results.

For the purposes of this section, it is convenient to use vector notation for subset sums. In particular, for a vector $x \in \mathbb{Z}^n$, we write $x \cdot w = \sum_{i=1}^{n} x_i w_i$, and $x^{-1}(j) \subseteq [n]$ for the set of $i \in [n]$ such that $v_i = j$.

Our approach to relate the number of sums $|w(2^{[n]})|$ to the largest bin size $\beta(w)$ is to establish a connection to the notion of Uniquely Decodable Code Pairs from information theory, defined as follows.

▶ **Definition 4.1** (Uniquely Decodable Code Pair, UDCP). If $A, B \subseteq \{0,1\}^n$ such that

$$|A + B| = |\{a + b : a \in A, b \in B\}| = |A| \cdot |B| ,$$

then (A, B) is called *uniquely decodable*. Note that here addition is performed over $\mathbb{Z}^n$ (and **not** mod $\mathbb{Z}_2^n$).

UDCP's capture the zero error region of the so-called *binary adder channel*, and there is a fair amount of work on how large the sets A and B can be (for a survey, see [19, §3.5.1]). The connection between UDCP's and Subset Sum is that a Subset Sum instance that both generates many sums and has a large bin yields a large UDCP, as captured in the following proposition.

▶ **Proposition 4.2.** *If there exist weights $w_1, \ldots, w_n$ such that $|w(2^{[n]})| = a$ and $\beta(w) = b$, then there exists a UDCP (A, B) with $|A| = a$ and $|B| = b$.*

Proof. Let $A \subseteq \{0,1\}^n$ be an injective set, i.e., $x \cdot w \neq x' \cdot w$ for all $x, x' \in A$ with $x \neq x'$. Note that there exists such an A with $|A| = a$. Let $B \subseteq \{0,1\}^n$ be a bin, i.e., $y \cdot w = y' \cdot w$ for all $y, y' \in B$. Note that we can take these to have sizes $|A| = a$ and $|B| = b$.

We claim that (A, B) is a UDCP. To see this, let $x, x' \in A$ and $y, y' \in B$ with $x+y = x'+y'$. Then

$$x \cdot w + y \cdot w = (x + y) \cdot w = (x' + y') \cdot w = x' \cdot w + y' \cdot w .$$

Thus $x \cdot w = x' \cdot w$, and so by the injectivity property of A, we have $x = x'$, which in turn implies $y = y'$ since $x + y = x' + y'$. ◀

We have the following result by Ordentlich and Shayevitz [17, Theorem 1, setting $R_1 = 0.997$ and $\alpha = 0.07$].

▶ **Theorem 4.3** ([17]). *Let $A, B \subseteq \{0,1\}^n$ such that (A, B) is a UDCP and $|A| \geq 2^{.997n}$. Then $|B| \leq 2^{0.4996n}$.*

With this connection in place, the proof of Lemma 1.3 is immediate.

▶ **Lemma 1.3** (restated). *If $|w(2^{[n]})| \geq 2^{0.997n}$, then $\beta(w) \leq 2^{0.4996n}$.*

Proof. Combine Theorem 4.3 with the contrapositive of Proposition 4.2. ◀

The remainder of this section is devoted to Lemma 1.4. The proof also (implictly) uses a connection to Uniquely Decodable Code Pairs, but here the involved sets of strings are not binary. There is no reason to believe that the constant 0.661 is tight. However, because a random instance w of density 2 satisfies the hypothesis for all partitions S, T and has $\beta(w) \approx 2^{0.5n}$ with good probability, just improving the constant 0.661 will not suffice for settling Open Question 1.

4.1 Proof of Lemma 1.4

For a subset $S \subseteq [n]$, define a function $b_S : \mathbb{Z} \to \mathbb{Z}$ by letting $b_S(x)$ be the number of subsets $S' \subseteq S$ such that $w(S') = x$. Note that $|w(2^S)|$ equals the support size of b_S, or $\|b_S\|_0$, and that $\beta_w(S) = \max_x b_S(x) = \|b_S\|_\infty$. Instead of working with these extremes, it is more convenient to work with the ℓ^2 norm of b_S, and the main technical claim to obtain Lemma 1.4 is the following.

► **Proposition 4.4.** *There exists a $\delta > 0$ such that for all sufficiently large $|S|$ the following holds: if $|w(2^S)| \geq 2^{(1-\delta)|S|}$, then $\|b_S\|_2 \leq 2^{0.661|S|}$.*

Proof of Proposition 4.4. Without loss of generality we take $S = [n]$, and to simplify notation we omit the subscript S from b_S and simply write $b : \mathbb{Z} \to \mathbb{Z}$ for the function such that $b(r)$ is the number of subsets of $w_1, \ldots, w_n$ summing to r. Note that

$$\|b\|_2^2 = \sum_{\substack{U,V\subseteq[n]\\ [w(U)=w(V)]}} 1 = \sum_{\substack{U,V\subseteq[n]\\ U\cap V=\emptyset\\ [w(U)=w(V)]}} 2^{n-|U|-|V|} = \sum_{y\in\{-1,0,1\}^n} [y \cdot w = 0] \cdot 2^{|y^{-1}(0)|},$$

where $[p]$ denotes 1 if p holds and 0 otherwise. Defining $B_\sigma = \{y \in \{-1,0,1\}^n : y \cdot w = 0 \text{ and } \|y\|_1 = \sigma n\}$, we thus have

$$\|b\|_2^2 = \sum_{i=0}^{n} |B_{i/n}| 2^{n-i} \leq n \max_\sigma |B_\sigma| 2^{(1-\sigma)n} \,. \tag{3}$$

We now proceed to bound the size of B_σ by an encoding argument. To this end, let $A \subseteq \{0,1\}^n$ be a maximal injective set of vectors. In other words, $|A| = |w(2^{[n]})| \geq 2^{0.99n}$, and for all pairs $x \neq x' \in A$, it holds that $x \cdot w \neq x' \cdot w$. We claim that $|A + B_\sigma| = |A| \cdot |B_\sigma|$. To see this note that, similarly to the proof of Proposition 4.2, if $x + y = x' + y'$ (with $x, x' \in A$ and $y, y' \in B_\sigma$) then $x \cdot w = x' \cdot w$ (since $y' \cdot w = 0$) and thus $x = x'$ and $y = y'$.

Define P_σ to be all pairs (x, y) in $A \times B_\sigma$ that are *balanced*, in the sense that for some $\gamma > 0$ the following conditions hold:

$$\begin{aligned} |x^{-1}(1) \cap y^{-1}(-1)| &= \tfrac{1}{2}|y^{-1}(-1)| \pm \gamma n \,, \\ |x^{-1}(1) \cap y^{-1}(0)| &= \tfrac{1}{2}|y^{-1}(0)| \pm \gamma n \,, \\ |x^{-1}(1) \cap y^{-1}(1)| &= \tfrac{1}{2}|y^{-1}(1)| \pm \gamma n \,. \end{aligned} \tag{4}$$

► **Claim 4.5.** *For $\gamma = \sqrt{\delta}$ and n sufficiently large, we have that $|P_\sigma| \geq |A| \cdot |B_\sigma|/2$.*

The postponed proof of Claim 4.5 can be found in the full version [3].

Setting $\gamma = \sqrt{\delta}$, we can now proceed to upper bound $|P_\sigma|$. Consider the encoding $\eta : P_\sigma \to \{-1, 0, 1, 2\}^n$ defined by $\eta(x, y) = x + y$. By the property $|A + B_\sigma| = |A| \cdot |B|$, it follows that η is an injection, and thus $|P_\sigma|$ equals the size of the image of η. For a pair $(x, y) \in P_\sigma$, if $y \in B_\sigma$ has $\tau\sigma n$ many 1's, and $(1-\tau)\sigma n$ many -1's, then $z = \eta(x, y)$ has the following frequency distribution:

$$\frac{|z^{-1}(-1)|}{n} = \frac{\tau\sigma}{2} \pm o_\gamma(1) \,, \qquad \frac{|z^{-1}(0)|}{n} = \frac{\tau\sigma}{2} + \frac{1-\sigma}{2} \pm o_\gamma(1) \,,$$

$$\frac{|z^{-1}(1)|}{n} = \frac{1-\sigma}{2} + \frac{(1-\tau)\sigma}{2} \pm o_\gamma(1) \,, \qquad \frac{|z^{-1}(2)|}{n} = \frac{(1-\tau)\sigma}{2} \pm o_\gamma(1) \,,$$

where, for a variable ϵ, we write $o_\epsilon(1)$ to indicate a term that converges to 0 when ϵ tends to 0. Since $\gamma = \sqrt{\delta}$, we have $o_\gamma(1) = o_\delta(1)$. The number of z's with such a frequency distribution is bounded by

$$\binom{n}{\frac{\tau\sigma}{2}n, (\frac{\tau\sigma}{2} + \frac{1-\sigma}{2})n, (\frac{1-\sigma}{2} + \frac{(1-\tau)\sigma}{2})n, \frac{(1-\tau)\sigma}{2}} 2^{o_\delta(1)n} . \tag{5}$$

Then, $|P_\sigma|$ is bounded by

$$\log|P_\sigma| \leq \max_{\tau\in[0,1]} \big(g(\sigma,\tau)+o_\gamma(1)\big)n, \text{ where } g(\sigma,\tau) = h\Big(\tfrac{\tau\sigma}{2}, \tfrac{\tau\sigma}{2}+\tfrac{1-\sigma}{2}, \tfrac{1-\sigma}{2}+\tfrac{(1-\tau)\sigma}{2}, \tfrac{(1-\tau)\sigma}{2}\Big) .$$

It can be verified that $g(\sigma,\tau)$ is maximized for $\tau = 1/2$ and we have

$$\max_{\tau\in[0,1]} g(\sigma,\tau) = h\big(\tfrac{\sigma}{4}, \tfrac{1}{2}-\tfrac{\sigma}{4}, \tfrac{1}{2}-\tfrac{\sigma}{4}, \tfrac{\sigma}{4}\big) = 1 + h\big(\tfrac{\sigma}{2}\big) .$$

Combining this with the bounds $|P_\sigma| \geq |A|\cdot|B|\cdot 2^{-O(\delta^2)n}$ and $|A| \geq 2^{(1-\delta)n}$, we get that $|B_\sigma| \leq 2^{(h(\sigma/2)+o_\delta(1))n}$. Plugging this into (3) we see that

$$\|b\|_2^2 \leq \max_\sigma 2^{(1+h(\sigma/2)-\sigma+o_\epsilon(1))n} .$$

The expression $h(\sigma/2) - \sigma$ is maximized at $\sigma = 2/5$, and we obtain

$$\|b\|_2^2 \leq 2^{(h(1/5)+3/5+o_\epsilon(1))} \leq 2^{(1.32195+o_\delta(1))n} .$$

Thus if δ is sufficiently small, we have $\|b\|_2^2 \leq 2^{1.322n}$, as desired. ◀

Using Proposition 4.4, the desired bound of Lemma 1.4 follows immediately, since

$$\beta([n]) = \max_{x\in\mathbb{Z}} \sum_{y\in\mathbb{Z}} b_S(y)b_T(x-y) \leq \max_{x\in\mathbb{Z}} \|b_S\|_2\|b_T\|_2 \leq 2^{0.661n} ,$$

where the first inequality is by Cauchy–Schwarz and the second inequality by Proposition 4.4.

5 Proof of Theorem 1.6

Proof of Theorem 1.6. Given oracle access to an algorithm that solves Subset Sum instance of density at least 1.003 in $O^*(2^{(0.5-\epsilon)n})$ time for some $\epsilon > 0$, we solve an arbitrary instance $w_1, w_2, \ldots, w_n, t$ of Subset Sum in time $O^*\big(2^{\max\{0.49991, 0.5-\epsilon\}n}\big)$ as follows.

As Step 1, run the algorithm of Theorem 1.5 for $\Theta^*(2^{0.49991n})$ timesteps. If it terminates within this number of steps, return YES if it found a solution and NO otherwise. Otherwise, as Step 2, run the preprocessing of Lemma 2.2 with $B = 10\cdot 2^{0.997n}$. This yields a new instance with density $1/0.997 > 1.003$, which we solve using the presumed oracle for such instances. If the oracle returns a solution, we verify that it is indeed a solution to our original instance and if so return YES. Otherwise we return NO.

If there is no solution this algorithm clearly returns NO. If there is a solution and $|w(2^{[n]})| \geq 2^{0.997n}$, we find a solution with inversely polynomial probability in Step 1. If there is a solution and $|w(2^{[n]})| \leq 2^{0.997n}$, Property 2 of Lemma 2.2 guarantees that the solution to the reduced instance is a solution to the original instance with probability at least polynomial in the input size, and the oracle will then provide us with the solution. ◀

6 Further Discussion

Our original ambition was to resolve Open Question 1 affirmatively by a combination of two algorithms that exploit small and large concentration of the sums, respectively. Since we only made some partial progress on this, it remains an intruiging question whether this approach can fulfill this ambition. In this section we speculate about some further directions to explore.

Exploiting Large Density. For exploiting a density $1.003 \leq d \leq 2$, the meet-in-the-middle technique [9] does not seem directly extendable. A different, potentially more applicable $O^*(2^{n/2})$ algorithm works as follows: pick a prime p of order $2^{n/2}$, build the dynamic programming table that counts the number of subsets with sum congruence to t mod p, and use this as a data structure to uniformly sample solutions mod p with linear delay; try $O^*(2^{n/2})$ samples and declare a no-instance if no true solution is found (see also Footnote 1). As such, this does not exploit large density at all, but to this end one could seek a similar sampler that is more biased to smaller bins.

Sharper Analysis of Algorithm 1. The analysis of Algorithm 1 in Lemma 3.1, and in particular the typical bottleneck $\beta(w)2^{(1.5-\gamma)\mu n}$ in the running time, is quite naive. For example, since we can pick M as we like (and assume it generates many sums), for the algorithm to fail we need an instance where big bins are encountered by the algorithm for many choices of M. It might be a good approach to first try to extend the set of instances that can be solved 'truly faster' in this way, e.g. to the set of all instances with $\beta(w) \leq 2^{(.5+\delta)n}$ for some small $\delta > 0$. As an illustration of the looseness, let us mention that in a previous version of this manuscript, we used a more sophisticated analysis to show the following: there exists some $\delta > 0$, such that if $|w(2^{[n]})| \geq 2^{(1-\delta)n}$, then $|\{(P,Q) \in \binom{[n]}{n/2}^2 : w(P)+w(Q)=t\}| \leq 2^{0.5254n}$. We used this to show that all instances with $|w(2^{[n]})| \geq 2^{(1-\delta)n}$ can be solved via a mild variant of Algorithm 1 with $M=[n]$, indicating that Algorithm 1 gives non-trivial algorithms even for large M.

Sharper Combinatorial Bounds. Lemma 1.3 and Lemma 1.4 seem to be rather crude estimates. In fact, we don't even know the following (again, borrowing notation from the proof of Proposition 4.4):

Open Question 3: Suppose $|w(2^{[n]})| \geq 2^{(1-\epsilon)n}$. Can $\beta(w)$ and $\|b_{[n]}\|_2$ be bounded by $2^{o_\epsilon(1)n}$ and $2^{(0.5+o_\epsilon(1))n}$, respectively?

Note that the second bound would follow from the first bound. Furthermore, if the second bound holds, we would be able to solve, for all $\epsilon > 0$, all instances with $|\beta(w)| \geq 2^{(0.5+\epsilon)n}$ in time $O^*(2^{(0.5-\epsilon')n})$ for some $\epsilon' > 0$ depending on ϵ, via the proof of Theorem 1.2.

In recent work [1] we proved the following modest progress:

▶ **Lemma 6.1.** *There exists $\delta > 0$ such that if $A, B \subseteq \{0,1\}^n$ is a UDCP and $|A| \geq 2^{(1-\delta)n}$, then $|B| \leq 2^{0.4115n}$.*

Plugging this into the proof of Lemma 1.3, this gives that $\beta(w) \leq 2^{(0.4115+o_\epsilon(1))n}$ in the setting of Open Question 3. We would like to remark that improving this beyond $2^{(0.25+o_\epsilon(1))n}$ via Lemma 1.3 is not possible since UDCP pairs (A,B) with $|A| \geq 2^{(1-o(1))n}$ and $|B| \geq 2^{n/4}$ do exist [12]. One may also wonder whether we can deal with instances with $|w(2^{[n]})| \geq 2^{(0.5+\epsilon)n}$, for all $\epsilon > 0$ by arguing $\beta(w)$ must be small but this does not work directly: there are instances with $|w(2^{[n]})| = 3^{n/2}$ and $\beta(w) = 2^{n/2}$ (the instance $1, 1, 3, 3, 9, 9, 27, 27, \ldots$ has this, though it is easily attacked via Lemma 3.2).

Acknowledgements. This work was funded by the 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 NWO VENI project 639.021.438 (J.N.).

References

1 Per Austrin, Petteri Kaski, Mikko Koivisto, and Jesper Nederlof. Improved uniquely decodable code pair bounds for unbalanced pairs. Unpublished.

2 Per Austrin, Petteri Kaski, Mikko Koivisto, and Jesper Nederlof. Subset sum in the absence of concentration. In Ernst W. Mayr and Nicolas Ollinger, editors, *32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, March 4-7, 2015, Garching, Germany*, volume 30 of *LIPIcs*, pages 48–61. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015. URL: `http://dx.doi.org/10.4230/LIPIcs.STACS.2015.48`, `doi:10.4230/LIPIcs.STACS.2015.48`.

3 Per Austrin, Mikko Koivisto, Petteri Kaski, and Jesper Nederlof. Dense subset sum may be the hardest. *CoRR*, abs/1508.06019, 2015. URL: `http://arxiv.org/abs/1508.06019`.

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 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.

7 Abraham Flaxman and Bartosz Przydatek. Solving medium-density subset sum problems in expected polynomial time. In Volker Diekert and Bruno Durand, editors, *STACS 2005, 22nd Annual Symposium on Theoretical Aspects of Computer Science, Stuttgart, Germany, February 24-26, 2005, Proceedings*, volume 3404 of *Lecture Notes in Computer Science*, pages 305–314. Springer, 2005. URL: `http://dx.doi.org/10.1007/978-3-540-31856-9_25`, `doi:10.1007/978-3-540-31856-9_25`.

8 Open problems for FPT school 2014. `http://fptschool.mimuw.edu.pl/opl.pdf`.

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

10 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.

11 Russell Impagliazzo and Moni Naor. Efficient cryptographic schemes provably as secure as subset sum. *Journal of Cryptology*, 9(4):199–216, 1996. URL: `http://dx.doi.org/10.1007/BF00189260`, `doi:10.1007/BF00189260`.

12 T. Kasami, Shu Lin, V.K. Wei, and Saburo Yamamura. Graph theoretic approaches to the code construction for the two-user multiple-access binary adder channel. *IEEE Transactions on Information Theory*, 29(1):114–130, 1983. `doi:10.1109/TIT.1983.1056614`.

13 Petteri Kaski, Mikko Koivisto, and Jesper Nederlof. Homomorphic hashing for sparse coefficient extraction. In Dimitrios M. Thilikos and Gerhard J. Woeginger, editors, *Parameterized*

and Exact Computation – 7th International Symposium, IPEC 2012, Ljubljana, Slovenia, September 12-14, 2012. Proceedings, volume 7535 of *Lecture Notes in Computer Science*, pages 147–158. Springer, 2012. URL: http://dx.doi.org/10.1007/978-3-642-33293-7_15, doi:10.1007/978-3-642-33293-7_15.

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

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

16 Jesper Nederlof, Erik Jan van Leeuwen, and Ruben van der Zwaan. Reducing a target interval to a few exact queries. In Branislav Rovan, Vladimiro Sassone, and Peter Widmayer, editors, *Mathematical Foundations of Computer Science 2012 – 37th International Symposium, MFCS 2012, Bratislava, Slovakia, August 27-31, 2012. Proceedings*, volume 7464 of *Lecture Notes in Computer Science*, pages 718–727. Springer, 2012. URL: http://dx.doi.org/10.1007/978-3-642-32589-2_62, doi:10.1007/978-3-642-32589-2_62.

17 Or Ordentlich and Ofer Shayevitz. A VC-dimension-based outer bound on the zero-error capacity of the binary adder channel. *CoRR*, abs/1412.8670, 2014. URL: http://arxiv.org/abs/1412.8670.

18 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.

19 Christian Sleger and Alex Grant. *Coordinated Multiuser Communications.* Springer, 2006.

20 Gerhard J. Woeginger. Open problems around exact algorithms. *Discrete Appl. Math.*, 156(3):397–405, 2008. URL: http://dx.doi.org/10.1016/j.dam.2007.03.023, doi:10.1016/j.dam.2007.03.023.