Equal-Subset-Sum Faster Than the Meet-in-the-Middle

Marcin Mucha* Jesper Nederlof† Jakub Pawlewicz‡ Karol Węgrzycki§

In the Equal-Subset-Sum problem, we are given a set S of n integers and the problem is to decide if there exist two disjoint nonempty subsets $A, B \subseteq S$, whose elements sum up to the same value. The problem is NP-complete. The state-of-the-art algorithm runs in $\mathcal{O}^*(3^{n/2}) \leq \mathcal{O}^*(1.7321^n)$ time and is based on the *meet-in-the-middle* technique. In this paper, we improve upon this algorithm and give $\mathcal{O}^*(1.7088^n)$ worst case Monte Carlo algorithm. This answers a question suggested by Woeginger in his inspirational survey.

Additionally, we analyse the polynomial space algorithm for Equal-Subset-Sum. A naive polynomial space algorithm for Equal-Subset-Sum runs in $\mathcal{O}^*(3^n)$ time. With read-only access to the exponentially many random bits, we show a randomized algorithm running in $\mathcal{O}^*(2.6817^n)$ time and polynomial space.

*Institute of Informatics, University of Warsaw, Poland, mucha@mimuw.edu.pl. Supported by project TOTAL that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 677651).

†Eindhoven University of Technology, The Netherlands, j.nederlof@tue.nl. Supported by the Netherlands Organization for Scientific Research under project no. 024.002.003 and the European Research Council under project no. 617951.

‡Institute of Informatics, University of Warsaw, Poland, pan@mimuw.edu.pl.

§Institute of Informatics, University of Warsaw, Poland, k.wegrzycki@mimuw.edu.pl. Supported by the grants 2016/21/N/ST6/01468 and 2018/28/T/ST6/00084 of the Polish National Science Center and project TOTAL that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 677651).

1 Introduction

In the Subset-Sum problem, we are given as input a set S of n integers $a_1, \dots, a_n$ and a target t. The task is to decide if there exists a subset of S, such that a total sum of the numbers in this subset is equal to t. This can be formulated in the following form:

$$\sum_{i=1}^{n} x_i a_i = t$$

and the task is to find $x_i \in \{0, 1\}$. Subset-Sum is one of the fundamental NP-complete problems. Study on the exact complexity of Subset-Sum led to the discovery of one of the most fundamental algorithmic tool: *meet-in-the-middle*. Horowitz and Sahni [24] used this technique to give a $\mathcal{O}^*(2^{n/2})$ algorithm for Subset-Sum in the following way: First, rewrite the Subset-Sum equation:

$$\sum_{i=1}^{\lfloor n/2 \rfloor} x_i a_i = t - \sum_{\lfloor n/2 \rfloor + 1}^{n} x_i a_i.$$

Then enumerate all $\mathcal{O}(2^{n/2})$ possible values of the left side $L(x_1, \dots, x_{\lfloor n/2 \rfloor})$ and $\mathcal{O}(2^{n/2})$ possible values of the right side $R(x_{\lfloor n/2 \rfloor + 1}, \dots, x_n)$. After that, it remains to look for the value that occurs in both L and R, i.e., *meeting* the tables L and R. One can do that efficiently by sorting (see [24] for details). To summarize, meet-in-the-middle technique is based on rewriting the formula as an equation between two functions and efficiently seeking any value that occurs in both of their images.

Later, Schroeppel and Shamir [38] observed that space usage of meet-in-the-middle can be improved to $\mathcal{O}^*(2^{n/4})$ by using space-efficient algorithm for 4-SUM. However, the time complexity remains unchallenged and one of the most prominent open problem in the area of exact algorithms is to improve upon *meet-in-the-middle* for Subset-Sum:

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

In this paper, we consider the Equal-Subset-Sum problem. We are given a set S of n integers and the task is to decide if there exist two disjoint nonempty subsets $A, B \subseteq S$, whose elements sum up to the same value. Similarly to Subset-Sum, this problem is NP-complete [44]. In the inspirational survey, Woeginger [43] noticed Equal-Subset-Sum can be solved by using meet-in-the-middle and asked if it can be improved: [1]

Open Question 2 (c.f., [42],[43]). *Can we improve upon the meet-in-the-middle algorithm for* Equal-Subset-Sum*?*

The folklore meet-in-the-middle algorithm for Equal-Subset-Sum (that we will present in the next paragraph) works in $\mathcal{O}^*(3^{n/2})$ time.

Folklore algorithm for Equal-Subset-Sum First, we arbitrarily partition S into $S_1 = \{a_1, \dots, a_{\lfloor n/2 \rfloor}\}$ and $S_2 = \{a_{\lfloor n/2 \rfloor + 1}, \dots, a_n\}$. Recall that in Equal-Subset-Sum we seek two subsets $A, B \subseteq S$, such that $A \cap B = \emptyset$ and $\Sigma(A) = \Sigma(B)$. We can write the solution as 4 subsets: $A_1 = A \cap S_1$, $A_2 = A \cap S_2$, $B_1 = B \cap S_1$ and $B_2 = B \cap S_2$, such that: $\Sigma(A_1) + \Sigma(A_2) = \Sigma(B_1) + \Sigma(B_2)$. In particular, it means that: $\Sigma(A_1) - \Sigma(B_1) = \Sigma(B_2) - \Sigma(A_2)$. So, the problem reduces to finding two vectors $x \in \{-1, 0, 1\}^{\lfloor n/2 \rfloor}$ and $y \in \{-1, 0, 1\}^{\lceil n/2 \rceil}$, such that:

[1] [42, 43] noticed that 4-SUM gives $\mathcal{O}^*(2^n)$ algorithm, but it actually gives a $\mathcal{O}^*(3^{n/2})$ algorithm, see Appendix C.

$$\sum_{i=1}^{\lfloor n/2 \rfloor} x_i a_i = \sum_{i=1}^{\lceil n/2 \rceil} y_i a_{i+\lfloor n/2 \rfloor}.$$

We can do this in $\mathcal{O}^*(3^{n/2})$ time as follows. First, enumerate and store all $3^{\lfloor n/2 \rfloor}$ possible values of the left side of the equation and all $3^{\lceil n/2 \rceil}$ possible values of the right side of the equation. Then look for a value that occurs in both tables (collision) in time $\mathcal{O}^*(3^{n/2})$ by sorting the values. The total running time is therefore $\mathcal{O}^*(3^{n/2})$. Analogously to Subset-Sum, one can improve the space usage of the above algorithm to $\mathcal{O}^*(3^{n/4})$ (see Appendix C).

A common pattern seems unavoidable in algorithms for Subset-Sum and Equal-Subset-Sum: we have to go through all possible values of the left and the right side of the equation. This enumeration dominates the time used to solve the problem. So, it was conceivable that perhaps no improvement for Equal-Subset-Sum could be obtained unless we improve an algorithm for Subset-Sum first [42, 43].

1.1 Our Contribution

While the meet-in-the-middle algorithm remains unchallenged for Subset-Sum, we show that, surprisingly, we can improve the algorithm for Equal-Subset-Sum. The main result of this paper is the following theorem.

Theorem 1.1. Equal-Subset-Sum *can be solved in* $\mathcal{O}^*(1.7088^n)$ *time with high probability.*

This positively answers Open Question 2. To prove this result we observe that the worst case for the meet-in-the-middle algorithm is that of a balanced solution, i.e., when $|A| = |B| = |S \setminus (A \cup B)| \approx n/3$. We propose a substantially different algorithm, that runs in $\mathcal{O}^*(2^{2/3n})$ time for that case. The crucial insight of the new approach is the fact that when $|A| \approx |B| \approx n/3$, then there is an abundance of pairs $X, Y \subseteq S$, $X \neq Y$ with $\Sigma(X) = \Sigma(Y)$. We use the *representation technique* to exploit this. Interestingly, that technique was initially developed to solve the average case Subset-Sum [9, 25].

Our second result is an improved algorithm for Equal-Subset-Sum running in polynomial space. The naive algorithm in polynomial space works in $\mathcal{O}^*(3^n)$ time by enumerating all possible disjoint pairs of subsets of S. This algorithm is analogous to the $\mathcal{O}^*(2^n)$ polynomial space algorithm for Subset-Sum. Recently, Bansal et al. [6] proposed a $\mathcal{O}^*(2^{0.86n})$ algorithm for Subset-Sum on the machine that has access to the exponential number of random bits. We show that a similar idea can be used for Equal-Subset-Sum.

Theorem 1.2. *There exists a Monte Carlo algorithm which solves* Equal-Subset-Sum *in polynomial space and time* $\mathcal{O}^*(2.6817^n)$. *The algorithm assumes random read-only access to exponentially many random bits.*

This result is interesting for two reasons. First, Bansal et al. [6] require nontrivial results in information theory. Our algorithm is relatively simple and does not need such techniques. Second, the approach of Bansal et al. [6] developed for Subset-Sum has a barrier, i.e., significantly new ideas must be introduced to get an algorithm running faster than $\mathcal{O}^*(2^{0.75n})$. In our case, this corresponds to the algorithm running in $\mathcal{O}^*(2^{1.5n}) \leq \mathcal{O}^*(2.8285^n)$ time and polynomial space (for elaboration see Section 4). We show that relatively simple observations about Equal-Subset-Sum enable us to give a slightly faster algorithm in polynomial space.

1.2 Related Work

The Equal-Subset-Sum was introduced by Woeginger and Yu [44] who showed that the problem is NP-complete. This reduction automatically excludes $2^{o(n)}$ algorithms for Equal-Subset-Sum assuming ETH (see Appendix B), hence for this problem we aspire to optimize the constant in the exponent. The best known constant comes from the meet-in-the-middle algorithm. Woeginger [43] asked if this algorithm for Equal-Subset-Sum can be improved.

Exact algorithms for Subset-Sum: Nederlof et al. [35] proved that in the exact setting Knapsack and Subset-Sum problems are equivalent.

Schroeppel and Shamir [38] showed that the meet-in-the-middle algorithm admits a time-space tradeoff, i.e., $\mathcal{T}\mathcal{S}^2 \leq \mathcal{O}^*(2^n)$, where $\mathcal{T}$ is the running time of the algorithm and $\mathcal{S} \leq \mathcal{O}^*(2^{n/2})$ is the space of an algorithm. This tradeoff was improved by Austrin et al. [2] for almost all tradeoff parameters.

Austrin et al. [3] considered Subset-Sum parametrized by the maximum bin size β and obtained algorithm running in time $\mathcal{O}^*(2^{0.3399n}\beta^4)$. Subsequently, Austrin et al. [4] showed that one can get a faster algorithm for Subset-Sum than meet-in-the-middle if $\beta \leq 2^{(0.5-\varepsilon)n}$ or $\beta \geq 2^{0.661n}$. In this paper, we use the hash function that is based on their ideas. Moreover, the ideas in [3, 4] were used in the recent breakthrough polynomial space algorithm [6] running in $\mathcal{O}^*(2^{0.86n})$ time.

From the pseudopolynomial algorithms perspective Knapsack and Subset-Sum admit $\mathcal{O}(nt)$ algorithm, where t is a value of a target. Recently, for Subset-Sum the pseudopolynomial algorithm was improved to run in deterministic $\widetilde{\mathcal{O}}(\sqrt{n}t)$ time by Koiliaris and Xu [29] and randomized $\widetilde{\mathcal{O}}(n+t)$ time by Bringmann [11] (and simplified, see [27, 30]). However, these algorithms have a drawback of running in pseudopolynomial space $\mathcal{O}^*(t)$. Surprisingly, Lokshtanov and Nederlof [32] presented an algorithm running in time $\widetilde{\mathcal{O}}(n^3t)$ and space $\widetilde{\mathcal{O}}(n^2)$ which was later improved to $\widetilde{\mathcal{O}}(nt)$ time and $\widetilde{\mathcal{O}}(n \log t)$ space assuming the Extended Riemann Hypothesis [11].

From a lower bounds perspective, no algorithm working in $\widetilde{\mathcal{O}}(\text{poly}(n)t^{0.99})$ exists for Subset-Sum assuming SETH or SetCover conjecture [18, 1].

Approximation: Woeginger and Yu [44] presented the approximation algorithm for Equal-Subset-Sum with the worst case ratio of 1.324. Bazgan et al. [7] considered a different formulation of approximation for Equal-Subset-Sum and showed an FPTAS for it.

Cryptography and the average case complexity: In 1978 Knapsack problems were introduced into cryptography by Merkle and Hellman [34]. They introduced a Knapsack based public key cryptosystem. Subsequently, their scheme was broken by using lattice reduction [39]. After that, many knapsack cryptosystems were broken with low-density attacks [31, 17].

More recently, Impagliazzo and Naor [26] introduced a cryptographic scheme that is provably as secure as Subset-Sum. They proposed a function $f(\overrightarrow{a}, S) = \overrightarrow{a}, \sum_{i\in S} a_i \pmod{2^{l(n)}}$, i.e., the function which concatenates $\overrightarrow{a}$ with the sum of the a_i's for $i \in S$. Function f is a mapping of an n bit string S to an $l(n)$ bit string and $\overrightarrow{a}$ are a fixed parameter. Our algorithms can be thought of as an attempt to find a collision of such a function in the worst case.

However, in the average case more efficient algorithms are known. Wagner [41] showed that when solving problems involving sums of elements from lists, one can obtain faster algorithms when there are many possible solutions. In the breakthrough paper, Howgrave-Graham and Joux [25] gave $\mathcal{O}^*(2^{0.337n})$ algorithm for an average case Subset-Sum. It was subsequently improved by Becker et al. [9] who gave an algorithm running in $\mathcal{O}^*(2^{0.291n})$. These papers introduced a *representation* technique that is a crucial ingredient in our proofs.

Total search problems: The *Number Balancing* problem is: given n real numbers $a_1, \ldots, a_n \in [0, 1]$, find two disjoint subsets $I, J \subseteq [n]$, such that the difference $|\sum_{i \in I} a_i - \sum_{j \in J} a_j|$ is minimized. The pigeonhole principle and the Chebyshev's inequality guarantee that there exists a solution with difference at most $\mathcal{O}(\frac{\sqrt{n}}{2^n})$. Karmarkar and Karp [28] showed that in polynomial time one can produce a solution with difference at most $n^{-\Theta(\log n)}$, but since then no further improvement is known.

Papadimitriou [36] considered the problem *Equal Sums*: given n positive integers such that their total sum is less than $2^n - 1$, find two subsets with the same sum. By the pigeonhole principle the solution always exists, hence the decision version of this problem is obviously in P. However the hard part is to actually find a solution. Equal Sums is in class PPP but it remains open to show that it is PPP-complete. Recently, this question gained some momentum. Hoberg et al. [23] showed that Number Balancing is as hard as *Minkowski*. Ban et al. [5] showed the reduction from Equal Sums to Minkowski and conjectured that Minkowski is complete for the class PPP. Very recently, Sotiraki et al. [40] identified the first natural problem complete for PPP.

In Appendix E we show that our techniques can also be used to solve Number Balancing for integers in $\mathcal{O}^*(1.7088^n)$ time.

Combinatorial Number Theory: If $\Sigma(S) < 2^n - 1$, then by the pigeonhole principle the answer to the decision version of Equal-Subset-Sum on S is always YES. In 1931 Paul Erdős was interested in the smallest maximum value of S, such that the answer to Equal-Subset-Sum on S is NO, i.e., he considered the function:

$$f(n) = \min\{\max\{S\} \mid \text{all subsets of } S \text{ are distinct},\ \ |S| = n,\ \ S \subseteq \mathbb{N}\}$$

and showed $f(n) > 2^n/(10\sqrt{n})$ [19]. The first nontrivial upper bound on f was $f(n) \leq 2^{n-2}$ (for sufficiently large n) [16]. Subsequently, Lunnon [33] proved that $f(n) \leq 0.2246 \cdot 2^n$ and Bohman [10] showed $f(n) \leq 0.22002 \cdot 2^n$. Erdős [20] offered 500 dollars for proof or disproof of conjecture that $f(n) \geq c2^n$ for some constant c.

Other Variants: Equal-Subset-Sum has some connections to the study of the structure of DNA molecules [14, 15, 12]. Cieliebak et al. [13] considered k-Equal-Subset-Sum, in which we need to find k disjoint subsets of a given set with the same sum. They obtained several algorithms that depend on certain restrictions of the sets (e.g., small cardinality of a solution). In the following work, Cieliebak et al. [15] considered other variants of Equal-Subset-Sum and proved their NP-hardness.

2 Preliminaries

Throughout the paper we use the $\mathcal{O}^*$ notation to hide factors polynomial in the input size and the $\widetilde{\mathcal{O}}$ notation to hide factors logarithmic in the input size. We also use $[n]$ to denote the set $\{1, \ldots, n\}$. If $S = \{a_1, \ldots, a_n\}$ is a set of integers and $X \subseteq \{1, \ldots, n\}$, then $\Sigma_S(X) := \sum_{i \in X} a_i$. Also, we use $\Sigma(S) = \sum_{s \in S} s$ to denote the sum of the elements of the set. We use the binomial coefficient notation for sets, i.e., for a set S the symbol $\binom{S}{k} = \{X \subseteq S \mid |S| = k\}$ is the set of all subsets of the set S of size exactly k.

We may assume that the input to Equal-Subset-Sum has the following properties:

- the input set $S = \{a_1, \ldots, a_n\}$ consists of positive integers,

- $\sum_{i=1}^{n} a_i < 2^{\tau n}$ for a constant $\tau < 10$,
- integer n is a multiple of 12.

These are standard assumptions for Subset-Sum (e.g., [3, 22]). For completeness, in Appendix A we prove how to apply reductions to Equal-Subset-Sum to ensure these properties.

We need the following theorem concerning the density of prime numbers [21, p. 371, Eq. (22.19.3)].

Lemma 2.1. *For a large enough integer b, there exist at least $2^b/b$ prime numbers in the interval $[2^b, 2^{b+1}]$.*

The binary entropy function is $h(\alpha) = -\alpha \log_2 \alpha - (1-\alpha)\log_2(1-\alpha)$ for $\alpha \in (0,1)$ and $h(0) = h(1) = 0$. For all integers $n \geq 1$ and $\alpha \in [0,1]$ such that σn is an integer, we have the following upper bound on the binomial coefficient [37]: $\binom{n}{\alpha n} \leq 2^{h(\alpha)n}$. We also need a standard bound on binary entropy function $h(x) \leq 2\sqrt{x(1-x)}$.

Throughout this paper all logarithms are base 2.

3 Faster Exponential Space Algorithm

In this section, we improve upon the meet-in-the-middle algorithm for Equal-Subset-Sum.

Theorem 3.1. Equal-Subset-Sum *can be solved in $\mathcal{O}^*(1.7088^n)$ time with high probability.*

Theorem 3.1 is proved by using two different algorithms for Equal-Subset-Sum. To bound the trade-off between these algorithms we introduce the concept of a *minimum solution*.

Definition 3.2 (Minimum Solution)**.** *For a set S of positive integers we say that a solution $A, B \subseteq S$ is a* minimum *solution if its* size *$|A| + |B|$ is smallest possible.*

We now assume that the size of the minimum solution has even size for simplicity of presentation. The algorithm and analysis for the case of odd-sized minimum solution is similar, but somewhat more messy due to all the floors and ceilings one needs to take care of.

In Section 3.1 we prove that the meet-in-the-middle approach for Equal-Subset-Sum already gives algorithm running in time $\mathcal{O}^*((3-\varepsilon)^{n/2})$ if the minimum solution A, B is *unbalanced*, i.e., $||A \cup B| - \frac{2n}{3}| > \varepsilon' n$ for some $\varepsilon' > 0$ depending on ε. Subsequently, in Section 3.2 we propose an algorithm for *balanced* instances, i.e., when the size of a minimum solution is close to 2/3. In particular, we show how to detect sets A, B with $\Sigma(A) = \Sigma(B)$ and $|A| \approx |B| \approx \frac{n}{3}$, with an $\mathcal{O}^*(2^{\frac{2}{3}n})$ time algorithm. By bounding trade-off between the algorithms from Section 3.1 and Section 3.2 we prove Theorem 3.1 and bound the running time numerically.

3.1 Equal-Subset-Sum for unbalanced solutions via meet-in-the-middle

Theorem 3.3. *If S is a set of n integers with a minimum solution of size ℓ, then* Equal-Subset-Sum *with input S can be solved in $\mathcal{O}^*(\binom{n/2}{\ell/2} 2^{\ell/2})$ time with high probability.*

Proof of Theorem 3.3. Algorithm 1 uses the meet-in-the-middle approach restricted to solutions of size ℓ. We will show that this algorithm solves Equal-Subset-Sum in the claimed running time.

The algorithm starts by randomly partitioning the set S into two equally sized sets S_1, S_2. Let A, B be a fixed minimum solution of size $|A \cup B| = \ell$. We will later show that with $\Omega(1/\text{poly}(n))$

Algorithm 1 UNBALANCEDEQUALSUBSETSUM(S, ℓ)

Randomly split S into two disjoint $S_1, S_2 \subseteq S$, such that $|S_1| = |S_2| = n/2$
Enumerate $C_1 = \{\Sigma(A_1) - \Sigma(B_1) \mid A_1, B_1 \subseteq S_1,\ A_1 \cap B_1 = \emptyset,\ |A_1| + |B_1| = \ell/2\}$
Enumerate $C_2 = \{\Sigma(A_2) - \Sigma(B_2) \mid A_2, B_2 \subseteq S_2,\ A_2 \cap B_2 = \emptyset,\ |A_2| + |B_2| = \ell/2\}$
if $\exists x_1 \in C_1, x_2 \in C_2$ such that $x_1 + x_2 = 0$ **then**
    Let $A_1, B_1 \subseteq S_1$ be such that $x_1 = \Sigma(A_1) - \Sigma(B_1)$
    Let $A_2, B_2 \subseteq S_2$ be such that $x_2 = \Sigma(A_2) - \Sigma(B_2)$
    return $(A_1 \cup A_2, B_1 \cup B_2)$
end if
return NO

probability $|(A \cup B) \cap S_1| = |(A \cup B) \cap S_2| = \ell/2$. We assume this is indeed the case and proceed with meet-in-the-middle. For S_1 we will list all A_1, B_1 that could possibly be equal to $S_1 \cap A$ and $S_1 \cap B$, i.e. disjoint and with total size $\ell/2$. We compute $x = \Sigma(A_1) - \Sigma(B_1)$ and store all these in C_1. We proceed analogously for S_2.

We then look for $x_1 \in C_1$ and $x_2 \in C_2$ such that $x_1 + x_2 = 0$. If we find it then we identify the sets A_1 and B_1 that correspond to x_1 and sets A_2 and B_2 that correspond to x_2 (the easiest way to do that is to store with each element of C_1 and C_2 the corresponding pair of sets when generating them). Finally we return $(A_1 \cup A_2, B_1 \cup B_2)$.

Probability of a good split: We now lower-bound the probability of S_1 and S_2 splitting $A \cup B$ in half. There are $\binom{n}{n/2}$ possible equally sized partitions. Among these there are $\binom{\ell}{\ell/2}\binom{n-\ell}{(n-\ell)/2}$ partitions that split $A \cup B$ in half. The probability that a random partition splits A and B in half is:

$$\frac{\binom{\ell}{\ell/2}\binom{n-\ell}{(n-\ell)/2}}{\binom{n}{n/2}} \geq \frac{2^{\ell}2^{n-\ell}}{(n+1)^2 2^n} = \frac{1}{(n+1)^2}$$

because $\frac{2^n}{n+1} \leq \binom{n}{n/2} \leq 2^n$.

Running time: To enumerate C_1 and C_2 we need $\mathcal{O}^*(\binom{n/2}{\ell/2}2^{\ell/2})$ time, because first we guess set $S_1 \cap (A \cup B)$ of size $\ell/2$ and then split between A and B in at most $2^{\ell/2}$ ways. We then check the existence of $x_1 \in C_1$ and $x_2 \in C_2$ such that $x_1 + x_2 = 0$ in $\mathcal{O}^*((|C_1| + |C_2|) \log(|C_1| + |C_2|))$ time by sorting.

We can amplify the probability of a *good split* to $\mathcal{O}(1)$ by repeating the whole algorithm polynomially many times.

Correctness: With probability $\Omega(1/\text{poly}(n))$ we divide the $A \cup B$ equally between S_1 and S_2. If that happens the set C_1 contains x_1 such that $x_1 = \Sigma(A \cap S_1) - \Sigma(B \cap S_1)$ and the set C_2 contains x_2 that $x_2 = \Sigma(A \cap S_2) - \Sigma(B \cap S_2)$. Note that $x_1 + x_2 = \Sigma(A \cap S_1) + \Sigma(A \cap S_2) - \Sigma(B \cap S_1) - \Sigma(B \cap S_2) = \Sigma(A) - \Sigma(B)$ which is 0, since A, B is a solution. Therefore Algorithm 1 finds a solution of size ℓ (but of course, it could be different from A,B). $\square$

3.2 EQUAL-SUBSET-SUM for balanced solutions

Theorem 3.4. *Given a set S of n integers with a minimum solution size $\ell \in (\frac{1}{2}n, (1-\varepsilon)n]$ for some constant $\varepsilon > 0$,* EQUAL-SUBSET-SUM *can be solved in time $\mathcal{O}^*(2^{\ell})$ w.h.p.*

We use Algorithm 2 to prove Theorem 3.4. In this algorithm, we first pick a random prime p in the range $[2^{n-\ell}, 2^{n-\ell+1}]$, as well as an integer t chosen uniformly at random from $[1, 2^{n-\ell}]$. We then compute the set $C = \{X \subseteq S \mid \Sigma(X) \equiv_p t\}$. In the analysis, we argue that with $\Omega(1/\mathrm{poly}(n))$ probability C contains two different subsets X, Y of S with $\Sigma(X) = \Sigma(Y)$. To identify such pair it is enough to sort the set $|C|$ in time $\mathcal{O}(|C| \log |C|)$, and then scan it. We return $X \setminus Y$ and $Y \setminus X$ to guarantee that the returned sets are disjoint.

Algorithm 2 BalancedEqualSubsetSum$(a_1, \ldots, a_n, \ell)$

1: Pick a random prime p in $[2^{n-\ell}, 2^{n-\ell+1}]$
2: Pick a random number t in $[1, 2^{n-\ell}]$
3: Let $C = \{X \subseteq S \mid \Sigma(X) \equiv_p t\}$ be the set of candidates ▷ C contains two sets with equal sum with probability $\Omega(1/\mathrm{poly}(n))$.
4: Enumerate and store all elements of C ▷ In time $\mathcal{O}^*(|C| + 2^{n/2})$
5: Find $X, Y \in C$, such that $\Sigma(X) = \Sigma(Y)$ ▷ In time $\mathcal{O}^*(|C|)$
6: **return** $(X \setminus Y, Y \setminus X)$

We now analyse the correctness of Algorithm 2. Later, we will give a bound on the running time and conclude the proof Theorem 3.4. First, observe the following:

Lemma 3.5. *Let S be a set of n positive integers with minimum solution size of ℓ. Let*

$$\Psi = \{\Sigma(X) \mid X \subseteq S \text{ and } \exists Y \subseteq S \text{ such that } X \neq Y \text{ and } \Sigma(X) = \Sigma(Y)\}\,. \tag{1}$$

If $\ell > \frac{n}{2}$, then $|\Psi| \geq 2^{n-\ell}$ (note that all elements in Ψ are different).

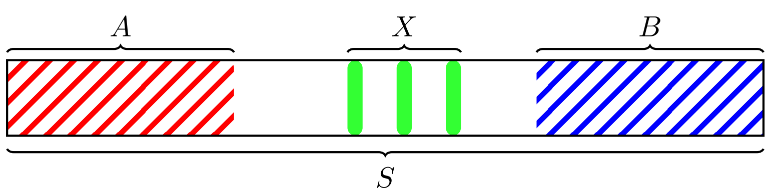

Figure 1: Scheme presents the set S of positive integers and two disjoint subsets $A, B \subseteq S$. The point is that if $\Sigma(A) = \Sigma(B)$ then for any subset $X \subseteq S \setminus (A \cup B)$ we have a guarantee that $\Sigma(A \cup X) = \Sigma(B \cup X)$.

Proof. Let $A, B \subseteq S$ be a fixed minimum solution to S. We know that $\ell = |A \cup B|$, $\Sigma(A) = \Sigma(B)$ and $A \cap B = \emptyset$. With this in hand we construct set Ψ of $2^{n-\ell}$ pairs of different $X, Y \subseteq S$ with $\Sigma(X) = \Sigma(Y)$.

Consider set $Z = S \setminus (A \cup B)$. By the bound on the size of A and B we know that $|Z| = n - \ell$. Now we construct our candidate pairs as follows: take any subset $Z' \subseteq Z$ and note that $X \cup Z'$ and $Y \cup Z'$ satisfy $\Sigma(X \cup Z') = \Sigma(Y \cup Z')$. There are $2^{|Z|}$ possible subsets of set Z and the claim follows.

Now we will prove that if $\ell > \frac{n}{2}$ then all subsets of Z have a different sum. Assume for a contradiction that there exist $Z_1, Z_2 \subseteq Z$, such that $\Sigma(Z_1) = \Sigma(Z_2)$ and $Z_1 \neq Z_2$. Then $Z_1 \setminus Z_2$ and $Z_2 \setminus Z_1$ would give a solution smaller than A, B, because $|Z| < \ell$. This contradicts the assumption about the minimality of A, B. It follows that if $\ell > \frac{1}{2}n$ then all constructed pairs have a different sum.

□

Now, we consider the hashing function $h_{t,p}(x) = x + t \pmod{p}$. We prove that if the set Ψ (see Equation 1) is sufficiently large, then for a random choice of t, at least one element of set Ψ is in the congruence class t.

Lemma 3.6. *Let S be the set of n positive integers bounded by $2^{\mathcal{O}(n)}$ with minimum solution of size ℓ and $\ell > \frac{n}{2}$. For a random prime $p \in [2^{n-\ell}, 2^{n-\ell+1}]$ and a random $t \in [1, 2^{n-\ell}]$ let $C_{t,p} = \{X \subseteq S \mid \Sigma(X) \equiv_p t\}$. Then,*

$$\mathbb{P}_{t,p}\left[\exists X, Y \in C_{t,p} \;\middle|\; \Sigma(X) = \Sigma(Y),\ X \neq Y\right] \geq \Omega(1/n^2).$$

Proof. Let Ψ be the set defined in (1). So $\Psi \subseteq \{1, \ldots, 2^{\mathcal{O}(n)}\}$, and $|\Psi| \geq 2^{n-\ell}$. It is sufficient to bound the probability, that there exists an element $a \in \Psi$ such $a \equiv_p t$. Let $a_1, a_2 \in \Psi$ be two distinct elements.

$$\mathbb{P}_p\left[a_1 \equiv_p a_2\right] = \mathbb{P}_p\left[p \text{ divides } |a_1 - a_2|\right] \leq \mathcal{O}(n(n-\ell)/2^{n-\ell}).$$

This is because $|a_1 - a_2|$ can only have $\mathcal{O}(n)$ prime divisors, and we are sampling p from the set of at least $2^{n-\ell}/(n-\ell)$ primes by Lemma 2.1. Let k be the number of pairs $a_1, a_2 \in \Psi$ such that $a_1 \equiv_p a_2$. We have $\mathbb{E}\left[k\right] \leq \mathcal{O}(|\Psi| + (|\Psi|n)^2/2^{n-\ell})$. We know that $|\Psi| \geq 2^{n-\ell}$, so $\frac{|\Psi|^2}{2^{n-\ell}} \geq |\Psi|$ which means that $\mathbb{E}\left[k\right] \leq \mathcal{O}((|\Psi|n)^2/2^{n-\ell})$. Hence, by Markov's inequality k is at most $\mathcal{O}((|\Psi|n)^2/2^{n-\ell})$ with at least constant probability. If this does indeed happen, then

$$|\{a \pmod{p} \mid a \in \Psi\}| \geq \frac{|\Psi|^2}{k} \geq \Omega\left(\frac{|\Psi|^2}{(|\Psi|n)^2/2^{n-\ell}}\right) \geq \Omega(2^{n-\ell}/n^2),$$

and the probability that t chosen uniformly at random from $[1, 2^{n-\ell}]$ will be among one of the elements of set $\{a \pmod{p} \mid a \in \Psi\}$ is $|\{a \pmod{p} \mid a \in \Psi\}|/2^{n-\ell} \geq \Omega(1/n^2)$.

□

Proof of correctness of Algorithm 2. By Lemma 3.6, after choosing a random prime p and random number $t \in [1, 2^{n-\ell}]$ the set $C = \{X \subseteq S \mid \Sigma(X) \equiv_p t\}$ contains at least two subsets $X, Y \subseteq S$, such that $\Sigma(X) = \Sigma(Y)$ with probability $\Omega(1/\text{poly}(n))$. Algorithm 2 computes the set C and finds $X', Y' \subseteq S$, such that $\Sigma(X') = \Sigma(Y')$. Then it returns the solution $X' \setminus Y', Y' \setminus X'$. □

Now we focus on bounding the running time of Algorithm 2. We start by bounding the size of the candidate set C.

Claim 3.7. *Let S be the set of n non-negative integers bounded by $2^{\mathcal{O}(n)}$ with a minimum solution of size ℓ such that $\ell \leq (1-\varepsilon)n$ for some constant $\varepsilon > 0$ (think of $\varepsilon = 1/100$). For a random prime $p \in [2^{n-\ell}, 2^{n-\ell+1}]$ and a random number $t \in [1, 2^{n-\ell}]$ let $C_{t,p} = \{X \subseteq S \mid \Sigma(X) \equiv_p t\}$. Then*

$$\mathbb{E}\left[|C_{t,p}|\right] \leq \mathcal{O}^*(2^{\ell})$$

Proof. By the linearity of expectations:

$$\mathbb{E}\left[|C_{t,p}|\right] = \sum_{X \subseteq S} \mathbb{P}_{t,p}\left[p \text{ divides } \Sigma(X) - t\right]$$

For the remaining part of the proof we focus on showing $\mathbb{P}_{t,p}\left[p \text{ divides } \Sigma(X) - t\right] \leq \mathcal{O}^*(2^{\ell-n})$ for a fixed $X \subseteq S$. It automatically finishes the proof, because there are 2^n possible subsets X.

We split the terms into two cases. If $\Sigma(X) = t$, then p divides $\Sigma(X) - t$ with probability 1. However, for a fixed $X \subseteq S$, the probability that $\Sigma(X) = t$ is $\mathcal{O}(\frac{1}{2^{n-\ell}})$ because t is a random number from $[1, 2^{n-\ell}]$ and $p \geq 2^{n-\ell}$.

On the other hand, if $\Sigma(X) \neq t$, then by the assumption, the set S consists of non-negative integers bounded by $2^{\tau n}$ for some constant $\tau > 0$. In particular, $|\Sigma(X) - t| \leq 2^{\tau n}$. This means that $|\Sigma(X) - t|$ has at most $\frac{\tau n}{n-\ell} \leq \frac{\tau}{\varepsilon} = \mathcal{O}(1)$ prime factors of size at least $2^{n-\ell}$. Any prime number p that divides $\Sigma(X) - t$ must therefore be one of these numbers. By Lemma 2.1 there are at least $2^{n-\ell}/(n-\ell)$ prime numbers in range $[2^{n-\ell}, 2^{n-\ell+1}]$. Hence, for a fixed $X \subseteq S$ the probability that p divides $\Sigma(X) - t$ is bounded by $\mathcal{O}(n2^{\ell-n})$. □

Lemma 3.8. *The set $C_{t,p}$ can be enumerated in time $\mathcal{O}^*\left(\max\left\{|C_{t,p}|, 2^{n/2}\right\}\right)$.*

The proof of the above lemma is based on Schroeppel and Shamir [38] algorithm for Subset-Sum. For a full proof of Lemma 3.8 see, e.g., Section 3.2 of [9]. Observe, that for our purposes the running time is dominated by $\mathcal{O}^*(|C_{t,p}|)$.

Proof of the running time of Algorithm 2. To enumerate the set $C_{t,p}$ we need $\mathcal{O}^*(|C| + 2^{n/2})$ time (see Lemma 3.8). To find two subsets $X, Y \in C$, such that $\Sigma(X) = \Sigma(Y)$ we need $\mathcal{O}^*(|C| \log |C|)$ time: we sort C and scan it.

The prime number p is at most $2^{n-\ell+1}$ and the expected size of C is $\mathcal{O}^*(2^\ell)$. Because we assumed that $\ell > \frac{n}{2}$ the expected running time is $\mathcal{O}^*(2^\ell)$ (we can terminate algorithm when it exceeds $\mathcal{O}^*(2^\ell)$ to Monte Carlo guarantees). The probability of success is $\Omega(1/\text{poly}(n))$. We can amplify it with polynomial overhead to any constant by repetition.

□

This concludes the proof of Theorem 3.4.

3.3 Trade-off for Equal-Subset-Sum

In this section, we will proof the Theorem 3.1 by combining Theorem 3.4 and Theorem 3.3.

Proof of Theorem 3.1. Both Theorem 3.4 and Theorem 3.3 solve Equal-Subset-Sum. Hence, we can focus on bounding the running time. By the trade-off between Theorem 3.4 (which works for $\ell \in (\frac{n}{2}, (1-\varepsilon)n)$ and Theorem 3.3 the running time is:

$$\mathcal{O}^*\left(\max_{\ell \in [1, n/2] \cup [(1-\varepsilon)n, n]} \left\{\binom{n/2}{\ell/2} 2^{\ell/2}\right\} + \max_{\ell \in (n/2, (1-\varepsilon)n)} \left\{\min\left\{\binom{n/2}{\ell/2} 2^{\ell/2}, 2^\ell\right\}\right\}\right)$$

For simplicity of analysis we bounded the sums by the maximum (note that $\mathcal{O}^*$ notation hides polynomial factors). When $\ell \leq n/2$, the running time is maximized for $\ell = n/2$, because (let $\ell = \alpha n$):

$$\mathcal{O}^*\left(\binom{n/2}{\ell/2} 2^{\ell/2}\right) = \mathcal{O}^*\left(2^{\frac{n}{2}(h(\alpha)+\alpha)}\right)$$

and the entropy function $h(x)$ is increasing in range $[0, 0.5)$. For $\ell = \frac{n}{2}$ the running time is $\mathcal{O}^*(2^{0.75n}) \leq \mathcal{O}^*(1.682^n)$. Similarly, we get a running time superior to the claimed one when $\ell \in [(1-\varepsilon)n, n]$. Note that $h(x) \leq 2\sqrt{x(1-x)}$, which means that the running time is bounded by $\mathcal{O}^*(2^{\frac{n}{2}(h(1-\varepsilon)+(1-\varepsilon))}) \leq \mathcal{O}^*(2^{\frac{n}{2}(1+2\sqrt{\varepsilon})})$ which is smaller than our running time for a sufficiently small constant ε.

Finally, when $\ell \in [n/2, (1-\varepsilon)n]$ we upper bound the running time by the:

$$\mathcal{O}^*\left(\max_{\ell\in[n/2,(1-\varepsilon)n)}\left\{\min\left\{2^{\frac{n}{2}(h(\alpha)+\alpha)}, 2^{\alpha n}\right\}\right\}\right).$$

The above expression is maximized when $h(\alpha)=\alpha$. By numeric calculations $\alpha < 0.77291$, which gives the final running time $\mathcal{O}^*(2^{\alpha n}) \leq \mathcal{O}^*(1.7088^n)$. ☐

4 Polynomial Space Algorithm

The naive algorithm for Equal-Subset-Sum in polynomial space works in $\mathcal{O}^*(3^n)$ time. We are given a set S. We guess a set $A \subseteq S$ and then guess a set $B \subseteq S \setminus A$. Finally, we check if $\Sigma(A) = \Sigma(B)$. The running time is:

$$\mathcal{O}^*\left(\binom{|S|}{|A|}\binom{|S|-|A|}{|B|}\right) \leq \mathcal{O}^*(3^n).$$

Known techniques for Subset-Sum allow us to get an algorithm running in $\mathcal{O}^*(2^{1.5n})$ and polynomial space.

Theorem 4.1. *There exists a Monte Carlo algorithm which solves* Equal-Subset-Sum *in polynomial space and $\mathcal{O}^*(2^{1.5n}) \leq \mathcal{O}^*(2.8285^n)$ time. The algorithm assumes random read-only access to exponentially many random bits.*

A crucial ingredient of Theorem 4.1 is a nontrivial result for the *Element Distinctness* problem [6, 8]. In this problem, one is given read-only access to the elements of a list $x \in [m]^n$ and the task is to find two different elements of the same value. The problem can be naively solved in $\mathcal{O}(n^2)$ time and $\mathcal{O}(1)$ space by brute force. Also by sorting, we can solve Element Distinctness in $\widetilde{\mathcal{O}}(n)$ time and $\widetilde{\mathcal{O}}(n)$ space. Beame et al. [8] showed that the problem can be solved in $\widetilde{\mathcal{O}}(n^{3/2})$ randomized time and $\widetilde{\mathcal{O}}(1)$ space. The algorithm assumes access to a random hash function $f : [m] \to [n]$.

Proof of Theorem 4.1. We can guarantee random access to the list $L = 2^S$ of all subsets of the set $S = \{a_1, \ldots, a_n\}$ on the fly. Namely, for a pointer $x \in \{0,1\}^n$ we can return an element of the list L that corresponds to x in $\mathcal{O}^*(1)$ time by choosing elements a_i for which $x_i = 1$. More precisely:

$$L(x_1, \ldots, x_n) = \{a_i \mid i \in [n],\ x_i = 1\}.$$

Now to decide Equal-Subset-Sum on set S we execute the Element Distinctness algorithm on the list L of sums of subsets. The list has size 2^n, hence the algorithm runs in $\mathcal{O}^*(2^{1.5n})$ time. Element Distinctness uses only polylogarithmic space in the size of the input, hence our algorithm uses polynomial space. ☐

Quite unexpectedly we can still improve upon this algorithm.

4.1 Improved Polynomial Space Algorithm

In this section, we show an improved algorithm.

Theorem 4.2. *There exists a Monte Carlo algorithm which solves* Equal-Subset-Sum *in polynomial space and time $\mathcal{O}^*(2.6817^n)$. The algorithm assumes random read-only access to exponentially many random bits.*

Similarly to the exponential space algorithm for EQUAL-SUBSET-SUM, we will combine two algorithms. We start with a generalization of Theorem 4.1 parametrized by the size of the solution.

Lemma 4.3. *Let S be a set of n positive integers, $A, B \subseteq S$ be the solution to* EQUAL-SUBSET-SUM *(denote $a = |A|$ and $b = |B|$). There exists a Monte Carlo algorithm which solves* EQUAL-SUBSET-SUM *in polynomial space and time*

$$\mathcal{O}^*\left(\left(\binom{n}{a} + \binom{n}{b}\right)^{1.5}\right).$$

The algorithm assumes random read-only access to exponentially many random bits.

Proof. The proof is just a repetition of the proof of Theorem 4.1 for a fixed sizes of solutions. Our list L will consists of all subsets $\binom{S}{a}$ and $\binom{S}{b}$. Then we run Element Distinctness algorithm, find any sets $A, B \in L$ such that $\Sigma(A) = \Sigma(B)$ and return $A \setminus B, B \setminus A$ to make them disjoint.

The running time follows because Element Distinctness runs in time $\widetilde{\mathcal{O}}(n^{1.5})$ and polylog(n) space. $\square$

Note that the runtime of Lemma 4.3 is maximized when $|A| = |B| = n/2$. The next algorithm gives improvement in that case.

Lemma 4.4. *Let S be a set of n positive integers, $A, B \subseteq S$ be the solution to* EQUAL-SUBSET-SUM *(denote $a = |A|$ and $b = |B|$). There exists a Monte Carlo algorithm which solves* EQUAL-SUBSET-SUM *in polynomial space and time*

$$\mathcal{O}^*\left(\min\left\{\binom{n}{a}2^{0.75(n-a)}, \binom{n}{b}2^{0.75(n-b)}\right\}\right).$$

The algorithm assumes random read-only access to exponentially many random bits.

Proof of Lemma 4.4. Without loss of generality, we focus on the case $a \leq b$. First we guess a solution set $A \subseteq S$. We answer YES if we find set $B \subseteq S \setminus A$ such that $\Sigma(A) = \Sigma(B)$ or find two disjoint subsets with equal sum in $S \setminus A$. We show that we can do it in $\mathcal{O}^*(2^{0.75(|S \setminus A|)})$ time and polynomial space which finishes the proof.

First, we arbitrarily partition set $S \setminus A$ into two equally sized sets S_1 and S_2. Then we create a list $L_1 = [\Sigma(X) \mid X \subseteq S_1]$ and list $L_2 = [\Sigma(A) - \Sigma(X) \mid X \subseteq S_2]$. We do not construct them explicitly because it would take exponential space. Instead we provide a read-only access to them (with the counter technique). We run Element Distinctness on concatenation of L_1 and L_2. If element distinctness found $x \in L_1$ and $y \in L_2$ such that $x = y$, then we backtrack and look for $X \subseteq S_1$, such that $\Sigma(X) = x$ and $Y \subseteq S_2$, such that $\Sigma(Y) = \Sigma(A) - y$ and return $(A, X \cup Y)$ which is a good solution, because $\Sigma(Y) + \Sigma(X) = \Sigma(A)$.

In the remaining case, i.e. when Element Distinctness finds a duplicate only in one of the lists then, we get a feasible solution as well. Namely, assume that Element Distinctness finds $x, y \in L_1$ such that $x = y$ (the case when $x, y \in L_2$ is analogous). Then we backtrack and look for two corresponding sets $X, Y \subseteq L_1$ such that $X \neq Y$ and $\Sigma(X) = \Sigma(Y) = x$. Finally we return $(X \setminus Y, Y \setminus X)$.

For the running time, note that the size of the list $|L_1| = |L_2| = 2^{0.5|S \setminus A|}$. Hence Element Distinctness runs in time $\mathcal{O}^*((|L_1| + |L_2|)^{1.5}) = \mathcal{O}^*(2^{0.75(n-a)})$. The backtracking takes time $\mathcal{O}^*(|L_1| + |L_2|)$ and polynomial space because we scan through all subsets of S_1 and all subsets of S_2 and look for a set with sum equal to the known value. $\square$

Proof of Theorem 4.2. By trade-off between Lemma 4.4 and Lemma 4.3 we get the following running time:

$$\mathcal{O}^*\left(\max_{1\le a,b\le n}\left\{\min\left\{\left(\binom{n}{a}+\binom{n}{b}\right)^{1.5},\binom{n}{a}2^{0.75(n-a)},\binom{n}{b}2^{0.75(n-b)}\right\}\right\}\right)$$

By symmetry this expression is maximized when $a = b$. Now we will write the exponents by using entropy function (let $a = \alpha n$):

$$\mathcal{O}^*\left(\max_{\alpha\in[0,1]}\left\{\min\left\{2^{1.5h(\alpha)n},2^{(h(\alpha)+0.75(1-\alpha))n}\right\}\right\}\right)$$

The expression is maximized when $1.5h(\alpha) = h(\alpha) + 0.75(1 - \alpha)$, By numerical computations $\alpha < 0.36751$, which means that the running time is $\mathcal{O}^*(2^{1.42312n}) \le \mathcal{O}^*(2.6817^n)$.

□

5 Conclusion and Open Problems

In this paper, we break two natural barriers for Equal-Subset-Sum: we propose an improvement upon the meet-in-the-middle algorithm and upon the polynomial space algorithm. Our techniques have additional applications in the problem of finding collision of hash function in cryptography and the number balancing problem (see Appendix E).

We believe that our algorithms can potentially be improved with more involved techniques. However, getting close to the running time of Subset-Sum seems ambitious. In Appendix B we show that a faster algorithm than $\mathcal{O}^*(1.1893^n)$ for Equal-Subset-Sum would yield a faster than $\mathcal{O}^*(2^{n/2})$ algorithm for Subset-Sum. It is quite far from our bound $\mathcal{O}^*(1.7088^n)$. The main open problem is therefore to close the gap between upper and lower bounds for Equal-Subset-Sum.

6 Acknowledgment

The authors would like to thank anonymous reviewers for their remarks and suggestions. This research has been initiated during Parameterized Algorithms Retreat of University of Warsaw 2019, Karpacz, 25.02-01.03.2019.

References

[1] A. Abboud, K. Bringmann, D. Hermelin, and D. Shabtay. Seth-based lower bounds for subset sum and bicriteria path. In T. M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 41–57. SIAM, 2019.

[2] P. Austrin, P. Kaski, M. Koivisto, and J. Määttä. Space-time tradeoffs for subset sum: An improved worst case algorithm. In F. V. Fomin, R. Freivalds, M. Z. Kwiatkowska, and D. Peleg, editors, *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part I*, volume 7965 of *Lecture Notes in Computer Science*, pages 45–56. Springer, 2013.

[3] P. Austrin, P. Kaski, M. Koivisto, and J. Nederlof. Subset sum in the absence of concentration. In E. W. Mayr and N. 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 fuer Informatik, 2015.

[4] P. Austrin, P. Kaski, M. Koivisto, and J. Nederlof. Dense subset sum may be the hardest. In N. Ollinger and H. Vollmer, editors, *33rd Symposium on Theoretical Aspects of Computer Science, STACS 2016, February 17-20, 2016, Orléans, France*, volume 47 of *LIPIcs*, pages 13:1–13:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

[5] F. Ban, K. Jain, C. H. Papadimitriou, C. Psomas, and A. Rubinstein. Reductions in PPP. *Inf. Process. Lett.*, 145:48–52, 2019.

[6] N. Bansal, S. Garg, J. Nederlof, and N. Vyas. Faster space-efficient algorithms for subset sum and k-sum. In H. Hatami, P. McKenzie, and V. King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 198–209. ACM, 2017.

[7] C. Bazgan, M. Santha, and Zs. Tuza. Efficient approximation algorithms for the subset-sums equality problem. In *International Colloquium on Automata, Languages, and Programming*, pages 387–396. Springer, 1998.

[8] P. Beame, R. Clifford, and W. Machmouchi. Element distinctness, frequency moments, and sliding windows. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 290–299. IEEE Computer Society, 2013.

[9] A. Becker, J. Coron, and A. Joux. Improved generic algorithms for hard knapsacks. In K. G. Paterson, editor, *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, volume 6632 of *Lecture Notes in Computer Science*, pages 364–385. Springer, 2011.

[10] T. Bohman. A sum packing problem of Erdős and the Conway-Guy sequence. *Proceedings of the American Mathematical Society*, 124(12):3627–3636, 1996.

[11] K. Bringmann. A near-linear pseudopolynomial time algorithm for subset sum. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '17, pages 1073–1084, Philadelphia, PA, USA, 2017. Society for Industrial and Applied Mathematics.

[12] M. Cieliebak. *Algorithms and hardness results for DNA physical mapping, protein identification, and related combinatorial problems*. PhD thesis, ETH Zürich, 2003.

[13] M. Cieliebak, S. Eidenbenz, and A. Pagourtzis. Composing equipotent teams. In *International Symposium on Fundamentals of Computation Theory*, pages 98–108. Springer, 2003.

[14] M. Cieliebak, S. Eidenbenz, and P. Penna. Noisy data make the partial digest problem np-hard. In *International Workshop on Algorithms in Bioinformatics*, pages 111–123. Springer, 2003.

[15] M. Cieliebak, S. Eidenbenz, A. Pagourtzis, and K. Schlude. On the complexity of variations of equal sum subsets. *Nord. J. Comput.*, 14(3):151–172, 2008.

[16] J. H. Conway and R. K. Guy. Sets of natural numbers with distinct subset sums. *Notices Amer. Math. Soc*, 15:345, 1968.

[17] M. J. Coster, A. Joux, B. A. LaMacchia, A. M. Odlyzko, C. Schnorr, and J. Stern. Improved low-density subset sum algorithms. *Computational Complexity*, 2:111–128, 1992.

[18] M. Cygan, H. Dell, D. Lokshtanov, D. Marx, J. Nederlof, Y. Okamoto, R. Paturi, S. Saurabh, and M. Wahlström. On problems as hard as CNF-SAT. In *Proceedings of the 27th Conference on Computational Complexity, CCC 2012, Porto, Portugal, June 26-29, 2012*, pages 74–84. IEEE Computer Society, 2012.

[19] P. Erdős. Problems and results in additive number theory. *Journal London Wash. Soc*, 16: 212–215, 1941.

[20] P. Erdős. A survey of problems in combinatorial number theory. *Annals of Discrete Mathematics*, 6:89–115, 1980.

[21] G. H. Hardy, E. M. Wright, et al. *An introduction to the theory of numbers.* Oxford university press, 1979.

[22] D. Harnik and M. Naor. On the compressibility of *NP* instances and cryptographic applications. *SIAM J. Comput.*, 39(5):1667–1713, 2010.

[23] R. Hoberg, H. Ramadas, T. Rothvoss, and X. Yang. Number balancing is as hard as Minkowski's theorem and shortest vector. 10328:254–266, 2017.

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

[25] N. Howgrave-Graham and A. Joux. New generic algorithms for hard knapsacks. In H. Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*, pages 235–256. Springer, 2010.

[26] R. Impagliazzo and M. Naor. Efficient cryptographic schemes provably as secure as subset sum. *J. Cryptology*, 9(4):199–216, 1996.

[27] C. Jin and H. Wu. A simple near-linear pseudopolynomial time randomized algorithm for subset sum. In J. T. Fineman and M. Mitzenmacher, editors, *2nd Symposium on Simplicity in Algorithms, SOSA@SODA 2019, January 8-9, 2019 - San Diego, CA, USA*, volume 69 of *OASICS*, pages 17:1–17:6. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2019.

[28] N. Karmarkar and R. M. Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 312–320. IEEE Computer Society, 1982.

[29] K. Koiliaris and C. Xu. A faster pseudopolynomial time algorithm for subset sum. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '17, pages 1062–1072, Philadelphia, PA, USA, 2017. Society for Industrial and Applied Mathematics.

[30] K. Koiliaris and C. Xu. Subset sum made simple. *CoRR*, abs/1807.08248, 2018.

[31] J. C. Lagarias and A. M. Odlyzko. Solving low-density subset sum problems. *J. ACM*, 32(1): 229–246, 1985.

[32] D. Lokshtanov and J. Nederlof. Saving space by algebraization. In L. J. Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 321–330. ACM, 2010.

[33] W. F. Lunnon. Integer sets with distinct subset-sums. *Mathematics of Computation*, 50(181): 297–320, 1988.

[34] R. C. Merkle and M. E. Hellman. Hiding information and signatures in trapdoor knapsacks. *IEEE Trans. Information Theory*, 24(5):525–530, 1978.

[35] J. Nederlof, E. J. van Leeuwen, and R. van der Zwaan. Reducing a target interval to a few exact queries. In B. Rovan, V. Sassone, and P. 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.

[36] C. H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *J. Comput. Syst. Sci.*, 48(3):498–532, 1994.

[37] H. Robbins. A remark on Stirling's formula. *The American mathematical monthly*, 62(1):26–29, 1955.

[38] R. Schroeppel and A. 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.

[39] A. Shamir. A polynomial-time algorithm for breaking the basic merkle-hellman cryptosystem. *IEEE Trans. Information Theory*, 30(5):699–704, 1984.

[40] K. Sotiraki, M. Zampetakis, and G. Zirdelis. Ppp-completeness with connections to cryptography. In M. Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 148–158. IEEE Computer Society, 2018.

[41] D. A. Wagner. A generalized birthday problem. In M. Yung, editor, *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, volume 2442 of *Lecture Notes in Computer Science*, pages 288–303. Springer, 2002.

[42] G. J. Woeginger. Space and time complexity of exact algorithms: Some open problems (invited talk). In R. G. Downey, M. R. Fellows, and F. K. H. A. Dehne, editors, *Parameterized and Exact Computation, First International Workshop, IWPEC 2004, Bergen, Norway, September 14-17, 2004, Proceedings*, volume 3162 of *Lecture Notes in Computer Science*, pages 281–290. Springer, 2004.

[43] G. J. Woeginger. Open problems around exact algorithms. *Discrete Applied Mathematics*, 156 (3):397–405, 2008.

[44] G. J. Woeginger and Z. Yu. On the equal-subset-sum problem. *Inf. Process. Lett.*, 42(6): 299–302, 1992.

A Preprocessing and Randomized Compression for EQUAL-SUBSET-SUM

We will repeat the arguments from [22]. Similar arguments are also present in [3, 4].

Theorem A.1. *Given set S of n integers $a_1, \dots, a_n \in \{-2^m, \dots, 2^m\}$ (with $m \gg n$). In $\mathcal{O}(poly(n, m))$ time we can construct a set S' that consists of n positive integers in $\{1, \dots, 2^{8n}\}$ such that:*

$$\exists X, Y \subseteq S, \text{ such that } \Sigma(X) = \Sigma(Y) \text{ iff } \exists X', Y' \subseteq S' \text{ such that } \Sigma(X') = \Sigma(Y')$$

with probability at least $1 - 2^{-n}$ or we will solve EQUAL-SUBSET-SUM *on that instance in polynomial time.*

Proof. If $0 \in S$, then we immediately answer YES, because sets $A = \{0\}$ and $B = \emptyset$ are a proper solution to EQUAL-SUBSET-SUM. If $m \geq 2^n$, then the algorithm running in time $\mathcal{O}(m4^n) \geq \mathcal{O}(m^3)$ runs in polynomial time of the instance size. Hence we can assume that $m < 2^n$ and $0 \notin S$.

Pick a random prime $p \in [2^{7n}, 2^{8n}]$. We will transform our original instance S into an instance S' in the following way:

$$a'_i \equiv a_i \pmod{p}$$

for all $i \in [n]$. In particular it means that all numbers in S' are positive and smaller than 2^{8n}. Observe, that if there is a solution for EQUAL-SUBSET-SUM on instance S, then the same set of indices is also a solution for EQUAL-SUBSET-SUM on instance S'. On the other hand, we want to show that if an answer to EQUAL-SUBSET-SUM on original instance S was NO, then for all pairs of subsets $A, B \subseteq S'$ it will hold that $\Sigma(A) \neq \Sigma(B)$.

For some $I, J \subseteq [n]$, in order to get $\sum_{i \in I} a'_i = \sum_{j \in J} a'_j$, while $\sum_{i \in I} a_i \neq \sum_{j \in J} a_j$, it must be that p is a divisor of $D(I, J) = \sum_{i \in I} a_i - \sum_{j \in J} a_j$. We will call such prime numbers *bad*.

There are 2^{2n} possible pairs of $I, J \subseteq [n]$. For a fixed $I, J \subseteq [n]$ there are at most $\log(n2^m)$ bad primes (because $D(I, J) \leq n2^m$). Hence there are at most:

$$2^{2n} \log(n2^m) \leq 2^{2n}(m + \log n) \leq 2^{4n}$$

possible bad primes. By Lemma 2.1, the prime number p is taken from the range containing at least 2^{7n} primes. Therefore, for every $I, J \subseteq [n]$ it holds that:

$$\mathbb{P}_p \left[\sum_{i \in I} a'_i = \sum_{j \in J} a'_j \right] \leq 2^{-3n}.$$

By talking union bound over all possible 2^{2n} pairs of $I, J \subseteq [n]$ the probability of error is bounded by 2^{-n}. □

What is left to prove, is that we can assume, that n is divisible by 12. By the above Lemma we know that S consists of only positive numbers. Let M be $\Sigma(S) + 1$. Observe that we can always add numbers from set $Z = \{M, 2M, 4M, 8M \dots, \}$ and the answer to EQUAL-SUBSET-SUM on the modified instance will not change because numbers in Z always have a different sum. Moreover, none of the subset of S can be used with numbers from Z, because $\Sigma(S) < M$. Hence we can always guarantee that n is divisible by 12 by adding appropriate amount of numbers from Z. Namely, note that if $n \equiv k \pmod{12}$, for some $k \neq 0$, then we can add k numbers to the original instance S and the answer to the EQUAL-SUBSET-SUM will not change.

B Sharper Reduction from Subset-Sum

In this section, we show a direct reduction from Subset-Sum. As far as we know, it is slightly sharper than currently known reduction [44] (in terms of constants in the exponent).

Theorem B.1. *If* Equal-Subset-Sum *can be solved in time* $\mathcal{O}^*((2-\varepsilon)^{0.25n})$ *for some* $\varepsilon > 0$ *then* Subset-Sum *can be solved in time* $\mathcal{O}^*((2-\varepsilon')^{0.5n})$ *for some constant* $\varepsilon' > 0$.

Proof. Assume that we have a black-box access to the algorithm for Equal-Subset-Sum running in time $\mathcal{O}^*((2-\varepsilon)^{0.25n})$ for some $\varepsilon > 0$. We will show how to use it to get an algorithm for Subset-Sum running in time $\mathcal{O}^*((2-\varepsilon)^{0.5n})$.

Given an instance S, t of Subset-Sum such that $S = \{a_1, \dots, a_n\}$, we will construct an equivalent instance S' of Equal-Subset-Sum such that $S' = \{s_1, \dots, s_{2n+1}\}$. Note, that for the running time this will be enough. The construction is as follows:

- for $1 \le i \le n$, let $s_i = a_i \cdot 10^{n+1} + 2 \cdot 10^i$,
- for $1 \le i \le n$, let $s_{i+n} = 1 \cdot 10^i$,
- let $s_{2n+1} = t \cdot 10^{n+1} + \sum_{i=1}^{n} 1 \cdot 10^i$.

First let us prove that if (S, t) is a YES instance of Subset-Sum then S' is a YES instance for Equal-Subset-Sum. Namely let $X \subseteq [n]$, such that $\sum_{i \in X} a_i = t$. Then, sets $A = \{s_i \mid i \in X\} \cup \{s_{i+n} \mid i \notin X\}$ and $B = \{s_{i+n} \mid i \in X\} \cup \{s_{2n+1}\}$ are a good solution to Equal-Subset-Sum on instance S', because $\sum(A) = \sum(B)$ and $A \cap B = \emptyset$.

Now for other direction, we will prove that if S' is a YES instance of Equal-Subset-Sum then (S, t) is a YES instance of Subset-Sum. Assume that S' is a YES instance and a pair $A, B \subseteq S'$ is a correct solution. Observe that if for some $i \le n$ element $s_i \in A$ then $s_{2n+1} \in B$. It is because the sets A, B have an equal sum and only elements s_i, s_{i+n}, s_{2n+1} have something nonzero at the i-th decimal place. Moreover all smaller decimal places of all numbers sum up to something smaller than 10^i and therefore cannot interfere with the place 10^i.

Finally observe, that numbers s_{i+n} for $i \in [n]$ cannot produce a YES instance on their own. Hence sets $A \cup B$ contain at least one number s_i for $i \in [n]$. WLOG let A be the set that contains such an s_i. Then set B has to contain s_{2n+1}. It means that set B cannot contain any s_i for $i \in [n]$.

In particular $\sum(A)/10^{n+1} = \sum(B)/10^{n+1}$. Only numbers s_i for $i \in [n]$ contribute to $\sum(A)/10^{n+1}$ and only number s_{2n+1} contributes to the $\sum(B)/10^{n+1}$. Hence there exists a subset $Z \subseteq S$, such that $\sum(Z) = t$. ☐

C Folklore Equal-Subset-Sum by 4-SUM with better memory

Theorem C.1. Equal-Subset-Sum *can be solved in deterministic* $\mathcal{O}^*(3^{n/2})$ *time and* $\mathcal{O}^*(3^{n/4})$ *space.*

Proof. First, we arbitrarily partition S into $S_1 = \{a_1, \dots, a_{n/4}\}$, $S_2 = \{a_{n/4+1}, \dots, a_{n/2}\}$ $S_3 = \{a_{n/2+1}, \dots, a_{3n/4}\}$ and $S_4 = \{a_{3n/4+1}, \dots, a_n\}$. Denote the vectors that correspond to these sets by $\overline{a}_1, \dots, \overline{a}_4 \in \mathbb{Z}^{n/4}$, i.e.,

$$\overline{a}_i = (a_{(i-1)n/4+1}, a_{(i-1)n/4+2}, \dots, a_{in/4}) \text{ for } i \in \{1, 2, 3, 4\}.$$

Recall that in Equal-Subset-Sum we were looking for two subsets $A, B \subseteq S$, such that $A \cap B = \emptyset$ and $\Sigma(A) = \Sigma(B)$. We can split the solution to 8 subsets:

$$A_i := S_i \cap A \text{ and } B_i := S_i \cap B \text{ for } i \in \{1,2,3,4\}.$$

Then, the equation for the solution is:

$$\Sigma(A_1) + \Sigma(A_2) + \Sigma(A_3) + \Sigma(A_4) = \Sigma(B_1) + \Sigma(B_2) + \Sigma(B_3) + \Sigma(B_4).$$

We can rewrite it as:

$$(\Sigma(A_1) - \Sigma(B_1)) + (\Sigma(A_2) - \Sigma(B_2)) + (\Sigma(A_3) - \Sigma(B_3)) + (\Sigma(A_4) - \Sigma(B_4)) = 0$$

Observe, that by definition $A_i \cap B_i = \emptyset$ for all $i \in \{1,2,3,4\}$. So the problem reduces to finding 4 vectors $x_1, x_2, x_3, x_4 \in \{-1,0,1\}^{n/4}$, such that:

$$\overline{a}_1 \cdot \overline{x}_1 + \overline{a}_2 \cdot \overline{x}_2 + \overline{a}_3 \cdot \overline{x}_3 + \overline{a}_4 \cdot \overline{x}_4 = 0. \tag{2}$$

because a term $\Sigma(A_i) - \Sigma(B_i)$ corresponds to $\overline{a}_i \cdot \overline{x}_i$ (1's from $\overline{x}_i$ correspond to the elements of A_i and -1's from $\overline{x}_i$ correspond to the elements of B_i).

Now, the algorithm is as follows. First enumerate all possible values of $\overline{a}_i \cdot \overline{x}_i$ for all $i \in \{1,2,3,4\}$ and store them in a table T_i. Along the way of value of $\overline{a}_i \cdot \overline{x}_i$ we store corresponding vector $\overline{x}$. Note, that $|T_i| = \mathcal{O}^*(3^{n/4})$. Now run 4-SUM on input tables T_i for $i \in \{1,2,3,4\}$, find $\overline{x}_i$ such that Equation (2) is satisfied. Then we find the corresponding sets A_i and B_i and return $(A_1 \cup A_2 \cup A_3 \cup A_4, B_1 \cup B_2 \cup B_3 \cup B_4)$. The 4-SUM finds vectors that sum to 0 from 4 different input sets. Because we enumerated all possibilities the correctness follows.

4-SUM runs in $\widetilde{\mathcal{O}}(|I|^2)$ time and $\widetilde{\mathcal{O}}(|I|)$ space where $|I|$ is the size of the input instance. In our case $|I| = \mathcal{O}^*(3^{n/4})$ and the running time and space complexity of the algorithm follows. ☐

D Time-Space Tradeoff for Equal-Subset-Sum

Schroeppel and Shamir [38] gave a time-space tradeoff for Subset-Sum, such that $\mathcal{T}\mathcal{S}^2 \leq \mathcal{O}^*(2^n)$ where $\mathcal{T}$ is a running time and $\mathcal{S}$ is the space of the algorithm for Subset-Sum and $\mathcal{S} \leq \mathcal{O}^*(2^{n/4})$. In this section we observe that similar relation is true for Equal-Subset-Sum:

Theorem D.1. *For all $\mathcal{S} \leq \mathcal{O}^*(3^{n/4})$,* Equal-Subset-Sum *can be solved in space $\mathcal{S}$ and time $\mathcal{T} \leq \mathcal{O}^*(\frac{3^n}{\mathcal{S}^2})$.*

Proof. Let S be the input instance of Equal-Subset-Sum and $\beta \in [0,1]$ be our trade-off parameter. By A, B we will denote a solution to Equal-Subset-Sum, i.e., $\Sigma(A) = \Sigma(B)$ and $A \cap B = \emptyset$.

Intuitively, for $\beta = 1$ we will use a polyspace algorithm running in time $\mathcal{O}^*(3^n)$ and for $\beta = 0$ we will use a meet in the middle algorithm running in $\mathcal{O}^*(3^{n/2}$ time and $\mathcal{O}^*(3^{n/4})$ space. First we arbitrarily choose a set X of βn elements of S. Then we guess set $A \cap X$ and set $B \cap X$. Finally we execute Equal-Subset-Sum meet-in-the-middle algorithm for an instance $(S \setminus X) \cup \{\Sigma(A \cap X), \Sigma(B \cap X)\}$ of $n(1-\beta)+2$ elements. The correctness follows because we checked all possible splits of X into sets A and B and put them into the solution. We did not increase possible solutions hence if the answer to Equal-Subset-Sum was NO then we will always answer NO. Similarly if the answer was YES, and the sets $A, B \subseteq S$ are a good solution, then for correctly guess $A \cap X$ and $B \cap X$ the constructed instance is a YES instance.

The algorithm runs in time $\mathcal{T}(n,\beta) = \mathcal{O}^*(3^{\beta n} \cdot \mathcal{T}(n(1-\beta)) \leq \mathcal{O}^*(3^{\beta n} 3^{(1-\beta)n/2})$ and space $\mathcal{S}(n,\beta) = \mathcal{O}^*(\mathcal{S}((1-\beta)n) \leq \mathcal{O}^*(3^{(1-\beta)n/4})$ (see Appendix C). It follows that:

$$\mathcal{T}(n,\beta)\mathcal{S}(n,\beta)^2 \leq \mathcal{O}^*(3^n)$$

Which gives us the final time-space tradeoff.

□

E Exact algorithm for Number Balancing

Recall, that in the *Number Balancing* problem you are given n real numbers $a_1, \dots, a_n \in [0, 1]$. The task is to find two disjoint subsets $I, J \subseteq [n]$, such that the difference $|\sum_{i \in I} a_i - \sum_{j \in J} a_j|$ is minimized. In this Section we show that our techniques transfer to the exact algorithm for Number Balancing. To alleviate problems with the definition of the computational model for real numbers, we will be solving the following problem:

Definition E.1 (Integer Number Balancing). *In the* Integer Number Balancing *problem, we are given a set S of n integers $a_1, \dots, a_n \in \{0, \dots, 2^{\mathcal{O}(n)}\}$. The task is to find two disjoint subsets $I, J \subseteq [n]$, such that the difference $\left|\sum_{i \in I} a_i - \sum_{j \in J} a_j\right|$ is minimized.*

Note, that Karmarkar and Karp [28] defined Number Balancing for reals because they were interested in approximation algorithms. For our purposes it is convenient to assume that numbers are given as integers bounded by $2^{\mathcal{O}(n)}$. For unbounded integers, some additional factors due to the arithmetic operations may occur.

Theorem E.2. *Integer Number Balancing can be solved in $\mathcal{O}^*(1.7088^n)$ time with high probability.*

It is convenient to work with the following decision version of the problem:

Definition E.3 (Integer Number Balancing, decision version). *In the decision version of* Integer Number Balancing*, we are given a set S of n integers $a_1, \dots, a_n \in \{0, \dots, 2^{\mathcal{O}(n)}\}$ and integer κ. The task is decide if there exist two disjoint subsets $I, J \subseteq [n]$, such that $\left|\sum_{i \in I} a_i - \sum_{j \in J} a_j\right| \in [0, \kappa]$.*

The above decision version and minimization version are equivalent up to polynomial factors: we use a binary search to for the smallest κ, for which answer to the decision version of Integer Number Balancing is YES. The target $\kappa \in [0, 2^{\mathcal{O}(n)}]$ so we need at most polynomial number of calls to the oracle.

E.1 Proof of Theorem E.2

First we observe, that our techniques also work for the generalization of Equal-Subset-Sum.

Definition E.4 (Target Equal-Subset-Sum problem). *In the* Target Equal-Subset-Sum *problem, we are given a set S of n integers and integer κ. The task is to decide if there exist two disjoint nonempty subsets $A, B \subseteq S$, such that $|\Sigma(A) - \Sigma(B)| = \kappa$.*

Theorem E.5. Target Equal-Subset-Sum *problem in $\mathcal{O}^*(1.7088^n)$ time with high probability.*

We give a sketch of the proof in Section E.2.

Now, we use an algorithm for Target Equal-Subset-Sum to give an algorithm for Integer Number Balancing. The observation is that decision version of Integer Number Balancing (see Definition E.3) asks if there exist two subsets $X, Y \subseteq S$ such that $|\Sigma(X) - \Sigma(Y)| \in [0, \kappa]$. However Theorem E.5 gives us an access to the oracle that determines if there exist two subsets $X, Y \subseteq S$, such that $|\Sigma(X) - \Sigma(Y)| = \kappa$. The following Lemma gives us tool for such a reduction:

Lemma E.6 (Shrinking Intervals, Theorem 1 from [35]). *Let U be a set of cardinality n, let $\omega : U \to \{-W, \dots, W\}$ be a weight function, and let $l < u$ be integers with $u - l > 1$. Then, there is a polynomial-time algorithm that returns a set of pairs $\Omega = \{(\omega_1, v_1), \dots, (\omega_T, v_T)\}$ with $\omega_i : U \to \{-W, \dots, W\}$ and integers $v_1, \dots, v_T \in \{-W, \dots, W\}$, such that:*

- *T is at most $\mathcal{O}(n \log(u - l))$, and:*
- *for every set $X \subseteq U$ it holds that $\omega(X) \in [l, u]$ if and only if there exist an index $i \in [T]$ such that $\omega_i(X) = v_i$.*

Note, that the corresponding Theorem in [35] was stated for weight function $\omega : U \to \{0, \dots, W\}$. However, the proof in [35] does not need that assumption. For clarity, in [35] weight functions $\omega_i : U \to \{-W, \dots, W\}$ are of the following form: for set $X \subseteq U$ the function is always $\omega_i(X) = \sum_{x \in X} w_x$ for some weights $w_i \in \mathbb{Z}$.

With Lemma E.6 in hand we can now prove Theorem E.2.

Proof of Theorem E.2. Let S be the set of n integers $\{a_1, \dots, a_n\}$ as in Definition E.3 and a target κ. Let $U = \{-n, \dots, -1\} \cup \{1, \dots, n\}$. For $z \in \mathbb{Z}$, let $\text{sgn}(z)$ be sign function, i.e., $\text{sgn}(z) = -1$ when $z < 0$, $\text{sgn}(0) = 0$ and $\text{sgn}(z) = 1$ when $z > 0$. Moreover, for any $X \subseteq U$ let $\omega(X) = \sum_{x \in X} \text{sgn}(x) a_{|x|}$.

We are given black-box access to the Theorem E.5, i.e., for a given set S' of integers we can decide if there exist two subsets $X, Y \subseteq S'$, such that $|\Sigma(X) - \Sigma(Y)| = \kappa$ in time $\mathcal{O}^*(1.7088^n)$. We show that we can solve Integer Number Balancing by using polynomial number of calls to Theorem E.5.

First, observe that universe set U and the weight function $\omega(X)$ satisfy the conditions of Lemma E.6. Moreover, let $u = \kappa$ and $l = -\kappa$. Lemma E.6 works in polynomial time and outputs pairs $P = \{(\omega_1, v_1), \dots, (\omega_T, v_T)\}$. Now, the answer to the decision version of Integer Number Balancing on S is YES iff there exists index $i \in [T]$ such that an answer to Target Equal-Subset-Sum on instance (ω_i, v_i) is YES by Lemma E.6.

For the running time observe, that the numbers are bounded by $2^{\mathcal{O}(n)}$, so $T = \mathcal{O}(\text{poly}(n))$. Hence, we execute polynomial number of calls to Theorem E.5 and the running time follows. ☐

E.2 Proof of Theorem E.5

What is left is to sketch that our techniques also apply to a more general version of the problem.

We are given a set S of n integers and a target κ. We need to find $X, Y \subseteq S$, such that $\Sigma(X) - \Sigma(Y) = \kappa$. First of all the definition of *minimum solution* for a target easily generalizes, i.e., we say that a solution $A, B \subseteq S$ such that $\Sigma(A) - \Sigma(B) = \kappa$ is a minimum solution if its size $|A| + |B|$ is smallest possible.

Note, that the meet-in-the-middle algorithm for Equal-Subset-Sum works for Target Equal-Subset-Sum (see Theorem 3.3 and Algorithm 1). The only difference is that in Algorithm 1, we need to determine if there exist $x_1 \in C_1, x_2 \in C_2$ such that $x_1 + x_2 = \kappa$. The running time and analysis is exactly the same in that case.

The main difference comes in the analysis of balanced case, i.e., Theorem 3.4. In that case we need to enumerate two sets $C_{t,p}$ and $C_{t-\kappa,p}$ (see Algorithm 3)

What is left is to show, that Algorithm 3 has the running time $\mathcal{O}^*(2^\ell)$ and finds a solution to Target Equal-Subset-Sum with probability $\Omega(1/\text{poly}(n))$. The rest of the proof and analysis is exactly the same as the proof of Theorem 3.4.

For the running time, note that $\mathbb{E}[|C_1|] \le \mathcal{O}^*(2^\ell)$ and $\mathbb{E}[|C_2|] \le \mathcal{O}^*(2^\ell)$ because these sets are chosen in exactly the same way as set C in Lemma 3.7. Moreover, we can enumerate both of them

Algorithm 3 BALANCEDEQUALSUBSETSUMTARGET$(a_1, \ldots, a_n, \ell, \kappa)$

1: Pick a random prime p in $[2^{n-\ell}, 2^{n-\ell+1}]$
2: Pick a random number $t \in [1, 2^{n-\ell}]$
3: Let $C_1 = \{X \subseteq S \mid \Sigma(X) \equiv_p t\}$
4: Let $C_2 = \{X \subseteq S \mid \Sigma(X) \equiv_p t - \kappa\}$
5: Enumerate and store all elements of C_1 and C_2 ▷ In time $\mathcal{O}^*(|C_1| + |C_2| + 2^{n/2})$
6: Find $X \in C_1$ and $Y \in C_2$, such that $\Sigma(X) - \Sigma(Y) = \kappa$ ▷ In time $\mathcal{O}^*(|C_1| + |C_2|)$
7: **return** $(X \setminus Y, Y \setminus X)$

in time $\mathcal{O}^*(|C_1| + |C_2| + 2^{n/2}) \leq \mathcal{O}^*(2^\ell)$ by using Lemma 3.8 (recall that $\ell > n/2$). Finally we can find $X \in C_1$ and $Y \in C_2$, such that $\Sigma(X) - \Sigma(Y) = \kappa$ (if such X, Y exist) in time $\mathcal{O}^*(2^\ell)$ by solving 2SUM. Hence, the running time of Algorithm 3 is $\mathcal{O}^*(2^\ell)$.

For the correctness, observe that an analog of Lemma 3.5 holds:

Lemma E.7. *Let S be a set of n positive integers with minimum solution size of ℓ. Let*

$$\Psi = \{\Sigma(X) \mid \exists Y \subseteq S \text{ such that } X \neq Y \text{ and } \Sigma(X) - \Sigma(Y) = \kappa\}\,. \tag{3}$$

If $\ell > \frac{n}{2}$, then $|\Psi| \geq 2^{n-\ell}$ (note that all elements in Ψ are different).

Proof of Lemma E.7. Similarly to the proof of Lemma E.7 we assume, that there exist $A, B \subseteq S$, such that $A \cap B = \emptyset$, $|A| + |B| = \ell$ and $\Sigma(A) - \Sigma(B) = t$. Then we construct our set Ψ be considering every subset $Z \subseteq S \setminus (A \cup B)$ and observing that:

- $\Sigma(A \cup Z) - \Sigma(B \cup Z) = \kappa$, and
- there are $2^{n-\ell}$ possible choices of set Z, and
- by the minimality of A, B all sets Z have a different sum.

□

And with that Lemma in hand we can prove the analogous to Lemma 3.6.

Lemma E.8. *Let S be the set of n positive integers bounded by $2^{\mathcal{O}(n)}$ with minimum solution A, B, $\Sigma(A) - \Sigma(B) = \kappa$ of size ℓ and $\ell > \frac{n}{2}$. For a random prime $p \in [2^{n-\ell}, 2^{n-\ell+1}]$ and a random $t \in [1, 2^{n-\ell}]$ let $C_{t,p} = \{X \subseteq S \mid \Sigma(X) \equiv_p t\}$. Then,*

$$\mathbb{P}_{t,p}\left[\exists X \subseteq C_{t,p}, Y \subseteq C_{t-\kappa,p} \;\middle|\; \Sigma(X) - \Sigma(Y) = \kappa,\ X \neq Y\right] \geq \Omega(1/n^2).$$

Proof of Lemma E.8. Recall Ψ from Lemma E.7. Note, that it is sufficient to show that there exist an element $a \in \Psi$, such that $a \equiv_p t$ with constant probability. Namely, if that is true, then $a \in C_{t,p}$ and by the definition of Ψ, there exists set $B \subseteq S$, such that $a - \Sigma(B) = \kappa$. Hence, $\Sigma(B) \in C_{t-\kappa,p}$ and the claim follows.

The rest of the proof, i.e., showing that $a \equiv_p t$ with constant probability is analogous to the proof of Lemma 3.6. □

With that in hand the correctness is analogous to the proof of correctness of Theorem 3.4.