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

Marcin Mucha
Institute of Informatics, University of Warsaw, Poland
mucha@mimuw.edu.pl

Jesper Nederlof
Eindhoven University of Technology, The Netherlands
j.nederlof@tue.nl

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

Karol Węgrzycki
Institute of Informatics, University of Warsaw, Poland
k.wegrzycki@mimuw.edu.pl

Abstract

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.

2012 ACM Subject Classification Mathematics of computing → Combinatorial algorithms

Keywords and phrases Equal-Subset-Sum, Subset-Sum, meet-in-the-middle, enumeration technique, randomized algorithm

Digital Object Identifier 10.4230/LIPIcs.ESA.2019.73

Related Version A full version of the paper [35] is available at `https://arxiv.org/abs/1905.02424`.

Funding *Marcin Mucha*: 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).
Jesper Nederlof: Supported by the Netherlands Organization for Scientific Research under project no. 024.002.003 and the European Research Council under project no. 617951.
Karol Węgrzycki: 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).

Acknowledgements 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.

1 Introduction

In the Subset-Sum problem, we are given as input a set S of n integers $a_1, \ldots, a_n$ 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*. [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 $L(x_1, \ldots, x_{\lfloor n/2 \rfloor})$ and $\mathcal{O}(2^{n/2})$ possible values of the right side $R(x_{\lfloor n/2 \rfloor + 1}, \ldots, x_n)$. 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, [39] 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 [45]. In the inspirational survey, [44] noticed Equal-Subset-Sum can be solved by using meet-in-the-middle and asked if it can be improved: [1]

▶ **Open Question 2** (cf., [43],[44])**.** *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 $S_1 = \{a_1, \ldots, a_{\lfloor n/2 \rfloor}\}$ and $S_2 = \{a_{\lfloor n/2 \rfloor + 1}, \ldots, a_n\}$. 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: $A_1 = A \cap S_1$, $A_2 = A \cap S_2$, $B_1 = B \cap S_1$

[1] [43, 44] noticed that 4-SUM gives $\mathcal{O}^*(2^n)$ algorithm, but it actually gives a $\mathcal{O}^*(3^{n/2})$ algorithm, see [35, Appendix C].

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:

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

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 [35, 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 [43, 44].

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, [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, [6] require nontrivial results in information theory. Our algorithm is relatively simple and does not need such techniques. Second, the approach of [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 [45] who showed that the problem is NP-complete. This reduction automatically excludes $2^{o(n)}$ algorithms for Equal-Subset-Sum assuming ETH (see [35, 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. [44] asked if this algorithm for Equal-Subset-Sum can be improved.

Exact algorithms for Subset-Sum

[36] proved that in the exact setting Knapsack and Subset-Sum problems are equivalent.

[39] 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 [2] for almost all tradeoff parameters.

[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, [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 [29] and randomized $\widetilde{\mathcal{O}}(n+t)$ time by [11] (and simplified, see [27, 30]). However, these algorithms have a drawback of running in pseudopolynomial space $\mathcal{O}^*(t)$. Surprisingly, [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

[45] presented the approximation algorithm for Equal-Subset-Sum with the worst case ratio of 1.324. [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 [34]. They introduced a Knapsack based public key cryptosystem. Subsequently, their scheme was broken by using lattice reduction [40]. After that, many knapsack cryptosystems were broken with low-density attacks [31, 17].

More recently, [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. [42] 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, [25] gave $\mathcal{O}^*(2^{0.337n})$

algorithm for an average case Subset-Sum. It was subsequently improved by [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 $\mathcal{O}(\frac{\sqrt{n}}{2^n})$. [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.

[37] 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. [23] showed that Number Balancing is as hard as *Minkowski*. [5] showed the reduction from Equal Sums to Minkowski and conjectured that Minkowski is complete for the class PPP. Very recently, [41] identified the first natural problem complete for PPP.

In [35, 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, [33] proved that $f(n) \leq 0.2246 \cdot 2^n$ and [10] showed $f(n) \leq 0.22002 \cdot 2^n$. [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 [15, 14, 12]. [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, [14] 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 [35, 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 [38]: $\binom{n}{\alpha n} \leq 2^{h(\alpha)n}$. 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.*

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

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 $\Omega(1/\text{poly}(n))$ 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:

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

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). ◀

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 $\Omega(1/\text{poly}(n))$ 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 $\Omega(1/\text{poly}(n))$.
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).

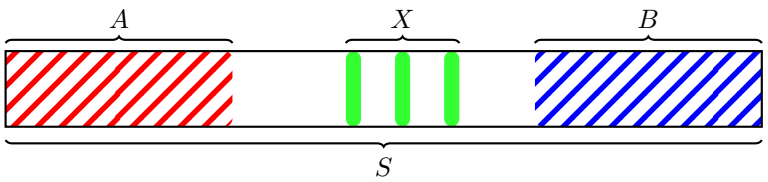

Figure 1 Scheme presents the set S of positive integers and two disjoint subsets $A, B \subseteq S$. The point is that if $\Sigma(A) = \Sigma(B)$ then for any subset $X \subseteq S \setminus (A \cup B)$ we have a guarantee that $\Sigma(A \cup X) = \Sigma(B \cup X)$.

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 $\mathbb{E}[k] \leq \mathcal{O}(|\Psi| + (|\Psi|n)^2/2^{n-\ell})$. We know that $|\Psi| \geq 2^{n-\ell}$, so $\frac{|\Psi|^2}{2^{n-\ell}} \geq |\Psi|$ which means that $\mathbb{E}[k] \leq \mathcal{O}((|\Psi|n)^2/2^{n-\ell})$. 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

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

Proof. By the linearity of expectations:

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

For the remaining part of the proof we focus on showing $\mathbb{P}_{t,p}[p \text{ divides } \Sigma(X) - t] \leq \mathcal{O}^*(2^{\ell-n})$ 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 [39] 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. [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. ◀

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. ◀

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

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

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 $\mathcal{O}^*(2^{1.42312n}) \leq \mathcal{O}^*(2.6817^n)$. ◀

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 [35, 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 [35, 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.

References

1. Amir Abboud, Karl Bringmann, Danny Hermelin, and Dvir Shabtay. SETH-Based Lower Bounds for Subset Sum and Bicriteria Path. In Timothy 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. Per Austrin, Petteri Kaski, Mikko Koivisto, and Jussi Määttä. Space-Time Tradeoffs for Subset Sum: An Improved Worst Case Algorithm. In Fedor V. Fomin, Rusins Freivalds, Marta Z. Kwiatkowska, and David Peleg, editors, *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. 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 fuer Informatik, 2015.
4. Per Austrin, Petteri Kaski, Mikko Koivisto, and Jesper Nederlof. Dense Subset Sum May Be the Hardest. In Nicolas Ollinger and Heribert 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. Frank Ban, Kamal Jain, Christos H. Papadimitriou, Christos-Alexandros Psomas, and Aviad Rubinstein. Reductions in PPP. *Inf. Process. Lett.*, 145:48–52, 2019.

6 Nikhil Bansal, Shashwat Garg, Jesper Nederlof, and Nikhil Vyas. Faster space-efficient algorithms for subset sum and k-sum. In Hamed Hatami, Pierre McKenzie, and Valerie 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 Cristina Bazgan, Miklos 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 Paul Beame, Raphaël Clifford, and Widad 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 Anja Becker, Jean-Sébastien Coron, and Antoine Joux. Improved Generic Algorithms for Hard Knapsacks. In Kenneth 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 Tom Bohman. A sum packing problem of Erdős and the Conway-Guy sequence. *Proceedings of the American Mathematical Society*, 124(12):3627–3636, 1996.

11 Karl 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 Mark Cieliebak. *Algorithms and hardness results for DNA physical mapping, protein identification, and related combinatorial problems.* PhD thesis, ETH Zürich, 2003.

13 Mark Cieliebak, Stephan Eidenbenz, and Aris Pagourtzis. Composing equipotent teams. In *International Symposium on Fundamentals of Computation Theory*, pages 98–108. Springer, 2003.

14 Mark Cieliebak, Stephan Eidenbenz, Aris Pagourtzis, and Konrad Schlude. On the Complexity of Variations of Equal Sum Subsets. *Nord. J. Comput.*, 14(3):151–172, 2008.

15 Mark Cieliebak, Stephan Eidenbenz, and Paolo Penna. Noisy data make the partial digest problem NP-hard. In *International Workshop on Algorithms in Bioinformatics*, pages 111–123. Springer, 2003.

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

17 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.

18 Marek Cygan, Holger Dell, Daniel Lokshtanov, Dániel Marx, Jesper Nederlof, Yoshio Okamoto, Ramamohan Paturi, Saket Saurabh, and Magnus 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 Paul Erdős. Problems and results in additive number theory. *Journal London Wash. Soc*, 16:212–215, 1941.

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

21 Godfrey Harold Hardy, Edward Maitland Wright, et al. *An introduction to the theory of numbers.* Oxford university press, 1979.

22 Danny Harnik and Moni Naor. On the Compressibility of *NP* Instances and Cryptographic Applications. *SIAM J. Comput.*, 39(5):1667–1713, 2010.

23 Rebecca Hoberg, Harishchandra Ramadas, Thomas Rothvoss, and Xin Yang. Number Balancing is as Hard as Minkowski's Theorem and Shortest Vector. In Friedrich Eisenbrand and Jochen Könemann, editors, *Integer Programming and Combinatorial Optimization - 19th International Conference, IPCO 2017, Waterloo, ON, Canada, June 26-28, 2017, Proceedings*, volume 10328 of *Lecture Notes in Computer Science*, pages 254–266. Springer, 2017.

24 Ellis Horowitz and Sartaj Sahni. Computing Partitions with Applications to the Knapsack Problem. *J. ACM*, 21(2):277–292, 1974.

25 Nick Howgrave-Graham and Antoine Joux. New Generic Algorithms for Hard Knapsacks. In Henri 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 Russell Impagliazzo and Moni Naor. Efficient Cryptographic Schemes Provably as Secure as Subset Sum. *J. Cryptology*, 9(4):199–216, 1996.

27 Ce Jin and Hongxun Wu. A Simple Near-Linear Pseudopolynomial Time Randomized Algorithm for Subset Sum. In Jeremy T. Fineman and Michael 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 Narendra Karmarkar and Richard 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 Konstantinos Koiliaris and Chao 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 Konstantinos Koiliaris and Chao Xu. Subset Sum Made Simple. *CoRR*, abs/1807.08248, 2018. `arXiv:1807.08248`.

31 J. C. Lagarias and Andrew M. Odlyzko. Solving Low-Density Subset Sum Problems. *J. ACM*, 32(1):229–246, 1985.

32 Daniel Lokshtanov and Jesper Nederlof. Saving space by algebraization. In Leonard 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 Fred Lunnon. Integer sets with distinct subset-sums. *Mathematics of Computation*, 50(181):297–320, 1988.

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

35 Marcin Mucha, Jesper Nederlof, Jakub Pawlewicz, and Karol Wegrzycki. Equal-Subset-Sum Faster Than the Meet-in-the-Middle. *CoRR*, abs/1905.02424, 2019. `arXiv:1905.02424`.

36 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.

37 Christos H. Papadimitriou. On the Complexity of the Parity Argument and Other Inefficient Proofs of Existence. *J. Comput. Syst. Sci.*, 48(3):498–532, 1994.

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

39 Richard Schroeppel and Adi Shamir. A T=O($2^{n/2}$). *SIAM J. Comput.*, 10(3):456–464, 1981.

40 Adi Shamir. A polynomial-time algorithm for breaking the basic Merkle-Hellman cryptosystem. *IEEE Trans. Information Theory*, 30(5):699–704, 1984.

41 Katerina Sotiraki, Manolis Zampetakis, and Giorgos Zirdelis. PPP-Completeness with Connections to Cryptography. In Mikkel 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.

42 David A. Wagner. A Generalized Birthday Problem. In Moti 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.

43 Gerhard J. Woeginger. Space and Time Complexity of Exact Algorithms: Some Open Problems (Invited Talk). In Rodney G. Downey, Michael R. Fellows, and Frank 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.

44 Gerhard J. Woeginger. Open problems around exact algorithms. *Discrete Applied Mathematics*, 156(3):397–405, 2008.

45 Gerhard J. Woeginger and Zhongliang Yu. On the Equal-Subset-Sum Problem. *Inf. Process. Lett.*, 42(6):299–302, 1992.