New Algorithms for Pigeonhole Equal Subset Sum

Ce Jin
MIT, Cambridge, MA, USA

Ryan Williams
MIT, Cambridge, MA, USA

Stan Zhang
MIT, Cambridge, MA, USA

Abstract

We study the *Pigeonhole Equal Subset Sum* problem, which is a total-search variant of the Subset Sum problem introduced by Papadimitriou (1994): we are given a set of n positive integers $\{w_1, \dots, w_n\}$ with the additional restriction that $\sum_{i=1}^{n} w_i < 2^n - 1$, and want to find two different subsets $A, B \subseteq [n]$ such that $\sum_{i \in A} w_i = \sum_{i \in B} w_i$.

Very recently, Jin and Wu (ICALP 2024) gave a randomized algorithm solving Pigeonhole Equal Subset Sum in $O^*(2^{0.4n})$ time, beating the classical meet-in-the-middle algorithm with $O^*(2^{n/2})$ runtime. In this paper, we refine Jin and Wu's techniques to improve the runtime even further to $O^*(2^{n/3})$.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases pigeonhole principle, subset sums

Digital Object Identifier 10.4230/LIPIcs.ESA.2025.84

Related Version This paper is based on Stan Zhang's Master's thesis [25].

Funding *Ce Jin*: Supported by the Jane Street Graduate Research Fellowship, NSF grant CCF-2330048, and a Simons Investigator Award.
Ryan Williams: Work supported in part by NSF CCF-2420092.
Stan Zhang: Self Funded

1 Introduction

Subset Sum is a fundamental NP-hard problem: given positive integers $w_1, w_2, \dots, w_n$ and a target integer t, we want to find a subset $A \subseteq [n]$ such that $\sum_{i \in A} w_i = t$.[1] Subset Sum has a simple meet-in-the-middle algorithm in about $O(2^{n/2})$ time, given by Horowitz and Sahni in 1974 [13]. There has been a long line of research attempting to improve that meet-in-the-middle running time for Subset Sum [4, 5, 22, 3, 14, 8, 9, 12], but it remains open whether there exists an algorithm running in $O(2^{(1/2-\varepsilon)n})$ time for any constant $\varepsilon > 0$. The fastest known algorithm for Subset Sum only has a poly(n)-factor improvement over $O(2^{n/2})$ [12].

Despite the lack of progress on the time complexity of Subset Sum, there has been significant progress for solving *variants* of the problem. Examples include Equal Subset Sum [16], 2-Subset Sum and Shifted Sums [2], more general subset balancing problems [11], algorithms solving either Equal Subset Sum or Subset Sum [20], Merlin–Arthur protocols for Subset Sum [17, 1], Subset Sum in low space [7, 18], and quantum algorithms for Subset Sum [2]. There is also a large body of works on pseudo-polynomial-time algorithms and approximation schemes for Subset Sum and related problems (see e.g., [10] and the references therein). In this work, we focus on exact exponential-time algorithms.

[1] Throughout this paper, we write $[n] = \{1, \dots, n\}$.

An important variant of Subset Sum is the Equal Subset Sum problem:

Equal Subset Sum [24]
Input: integers $w_1, w_2, \ldots, w_n$ where $|w_i| \le 2^{\mathrm{poly}(n)}$
Output: two different subsets $A, B \subseteq [n]$ such that $\sum_{i\in A} w_i = \sum_{i \in B} w_i$ (if exist).

Woeginger and Yu [24] first studied the Equal Subset Sum problem and showed that it is NP-Complete, similarly to Subset Sum. Since we can assume the solution satisfies $A \cap B = \emptyset$ (otherwise, we can return $(A \setminus B, B \setminus A)$ instead), a standard meet-in-the-middle approach can solve Equal Subset Sum in $O^*(3^{n/2}) \le O^*(1.7321^n)$ time. Mucha, Nederlof, Pawlewicz, and Węgrzycki [16] gave a faster $O^*(1.7088^n)$-time algorithm in 2019, answering an open question of [23].[2]

A special case of the Equal Subset Sum problem is the *Pigeonhole Equal Subset Sum* problem (also sometimes called Pigeonhole Equal Sums). This problem is the focus of our paper.

Pigeonhole Equal Subset Sum [19]
Input: positive integers $w_1, w_2, \ldots, w_n$, with promise $\sum_{i=1}^n w_i < 2^n - 1$.
Output: two different subsets $A, B \subseteq [n]$ such that $\sum_{i\in A} w_i = \sum_{i \in B} w_i$.

Pigeonhole Equal Subset Sum is a *total search problem*, in that there always exists a solution; our goal is merely to find one. To see why this is true, note that there are 2^n subsets $S \subseteq [n]$, which can only attain $2^n - 1$ possible subset sums $\sum_{i \in S} w_i \in \{0, 1, \ldots, 2^n - 2\}$ due to the promise $\sum_{i=1}^n w_i < 2^n - 1$, so by the pigeonhole principle there must exist a pair of subsets with the same subset sum.

The Pigeonhole Equal Subset Sum problem was introduced by Papadimitriou in 1994 [19], and has been studied in the TFNP literature [6, 21]. It belongs to the total search complexity class PPP, and is conjectured to be PPP-complete [19].

Pigeonhole Equal Subset Sum can be solved faster than the best known Equal Subset Sum algorithm. Until very recently, the fastest known algorithm for Pigeonhole Equal Subset Sum had time complexity $O^*(2^{n/2})$, either by a simple binary search combined with the classical meet-in-the-middle approach by Horowitz and Sahni [13] (see a short description in [15, Section 2]), or by a mod-p dynamic programming algorithm by Allcock, Hamoudi, Joux, Klingelhöfer, and Santha [2, Theorem 6.2]. Recently, a major advance by Jin and Wu [15] gave a randomized algorithm for Pigeonhole Equal Subset Sum running in $O^*(2^{0.4n})$ time. Roughly speaking, the key idea of Jin and Wu was to show that a Pigeonhole Equal Subset Sum instance with very few solutions must satisfy a certain structural property: namely, the input numbers should resemble the geometric progression $1, 2, 4, \ldots, 2^{n-1}$ (see the formal statement in Lemma 4).

1.1 Our contribution

In this paper, we improve the time complexity of Jin and Wu [15]'s Pigeonhole Equal Subset Sum algorithm from $O^*(2^{0.4n})$ to the clean bound $O^*(2^{n/3})$:

▶ **Theorem 1** (Main). *Pigeonhole Equal Subset Sum can be solved by a randomized algorithm in $O^*(2^{n/3})$ time.*

[2] Throughout this paper, we use $O^*(\cdot), \Omega^*(\cdot)$ to hide poly(n) factors, where n is the number of input integers.

Jin and Wu used similar techniques to give a *polynomial-space* algorithm for Pigeonhole Equal Subset Sum in $O^*(2^{0.75n})$ time, improving the straightforward $O^*(2^n)$-time polynomial-space algorithm based on binary search and meet-in-the-middle. We show that their polynomial-space algorithm can also be further improved:

▶ **Theorem 2.** *Pigeonhole Equal Subset Sum can be solved by a randomized algorithm in* $O^*(2^{2n/3})$ *time and* $\operatorname{poly}(n)$ *space.*

Our improved algorithms are based on the same structural property (see the formal statement in Lemma 4) proved by Jin and Wu [15], but we exploit it in a more efficient way. Jin and Wu [15] only exploited the property in the case when there are few solutions; if there are many solutions, they simply switched to a subsampling approach to try to hit a solution randomly, without using the structural property. In contrast, we make use of the structural property even in the case where there are many solutions, thereby improving the overall time complexity.

The rest of the paper is organized as follows: In Section 2, we give an overview of Jin and Wu's algorithm, describe their aforementioned structural property precisely, and extract a few lemmas from their work which are useful to us. Then in Section 3, we present our improved algorithm for Pigeonhole Equal Subset Sum, proving Theorem 1. In Section 4 we prove Theorem 2. Finally, we discuss some future research directions in Section 5.

2 A Summary of Jin and Wu's Algorithm

Throughout this paper, we use $w_1, \dots, w_n$ to denote the input integers, and we use the shorthand $w(A) = \sum_{i \in A} w_i$ for the subset sum attained by the subset of indices $A \subseteq [n]$.

For an Equal Subset Sum instance $(w_1, \dots, w_n)$ (not necessarily satisfying the pigeonhole promise), Jin and Wu [15] considered the parameter $F(w_1, \dots, w_n)$ defined as follows:

▶ **Definition 3** ($f_t(\cdot)$ **and** $F(\cdot)$)**.** *For* $w_1, \dots, w_n \in \mathbb{Z}$ *and* $t \in \mathbb{Z}$*, let*

$$f_t(w_1, \dots, w_n) := |\{S \subseteq [n] : w(S) = t\}|,$$

be the number of ways to achieve subset sum t*. When* $w_1, \dots, w_n$ *are clear from the context, we abbreviate* $f_t = f_t(w_1, \dots, w_n)$*.*

Then, define

$$F(w_1, \dots, w_n) := \sum_{t \in \mathbb{Z}} \max\{0, f_t - 1\}.$$

Observe that $F(w_1, \dots, w_n) \in \{0, 1, \dots, 2^n - 1\}$*.*

Note that an Equal Subset Sum instance $(w_1, \dots, w_n)$ has a solution $A, B \subseteq [n], A \neq B, w(A) = w(B)$ if and only if $F(w_1, \dots, w_n) \geq 1$.

Suppose the Pigeonhole Equal Subset Sum input instance consists of positive integers $w_1 < w_2 < \cdots < w_n$ (assuming no trivial solution $w_i = w_j, i \neq j$ exists) where $w([n]) < 2^n - 1$ (recall that this upper bound on the sum of all weights is the promise given by Pigeonhole Equal Subset Sum). Jin and Wu's key structural property relies on the following extra condition [15, Equation (1)]:

$$w([i]) \geq 2^i - 1, \text{ for all } i \in [n-1]. \tag{1}$$

That is, Equation (1) says that the total weight of the set $\{1, \dots, i\}$ is at least $2^i - 1$. We may assume without loss of generality that Equation (1) holds. To see why, note that if

$w([i]) < 2^i - 1$ for some $i \le n-1$, then $\{w_1, \dots, w_i\}$ is also a Pigeonhole Equal Subset Sum instance. We can solve this strictly smaller instance instead, and obtain a solution $A, B \subseteq [i] \subsetneq [n], A \neq B, w(A) = w(B)$. Hence, we may assume such i does not exist, so Equation (1) holds.

Using Equation (1), Jin and Wu [15] proved a key structural lemma for Pigeonhole Equal Subset Sum instances $(w_1, \dots, w_n)$. Their lemma says that if $F(w_1, \dots, w_n)$ is small, then the sequence $w_1, \dots, w_n$ is close to the geometric progression $1, 2, 4, \dots, 2^{n-1}$ with small additive error.

▶ **Lemma 4** ([15, Equation (8)]). *Suppose positive integers $w_1 < w_2 < \cdots < w_n$ satisfy $w([n]) < 2^n - 1$ and Equation (1). Then, for all $i \in [n]$,*

$$-i \cdot F(w_1, \dots, w_n) \le w_i - 2^{i-1} \le F(w_1, \dots, w_n). \tag{2}$$

Lemma 4 will be used in our improved algorithm as well. For completeness, in Appendix A we include a short proof of Lemma 4 given by Jin and Wu [15]. Jin and Wu gave two different algorithms, which we summarize below in Lemma 5 and Lemma 6. The first algorithm is suitable when the Pigeonhole Equal Subset Sum instance $(w_1, \dots, w_n)$ has small $F(w_1, \dots, w_n)$. It was obtained by using Lemma 4 to speed up the standard binary search and meet-in-the-middle approach:

▶ **Lemma 5** ([15, Lemma 4]). *Given parameter $1 \le \Delta \le 2^n/(3n^2)$, a Pigeonhole Equal Subset Sum instance $(w_1, \dots, w_n)$ (where $w_i < w_{i+1}$) satisfying $F(w_1, \dots, w_n) \le \Delta$ and Equation (1) can be solved deterministically in $O^*(\sqrt{\Delta})$ time.*

The second algorithm of Jin and Wu works well when $F(w_1, \dots, w_n)$ is large. The intuition is that, in this case there are many solutions, so one can perform subsampling and still expect at least one solution to survive. Jin and Wu used this subsampling idea to speed up the mod-p dynamic programming algorithm (which had previously been used for Subset Sum and related problems, e.g., [2, 4]):

▶ **Lemma 6** ([15, Lemma 5]). *Given parameter $2^{n/2} \le \Delta < 2^n$, an Equal Subset Sum instance $(w_1, \dots, w_n)$ with $F(w_1, \dots, w_n) \ge \Delta$ can be solved in $O^*((2^{2n}/\Delta)^{1/3})$ time by a randomized algorithm.*

We remark that originally [15, Lemma 5] was stated for Pigeonhole Equal Subset Sum only. However, upon inspecting their proof, one can easily verify that the pigeonhole promise was never used, so that the algorithm actually works for Equal Subset Sum as well. This point turns out to be useful for our improvement later.

Jin and Wu's final algorithm for Pigeonhole Equal Subset Sum combines Lemma 5 and Lemma 6, and the overall runtime is balanced at the bottleneck case $\Delta \approx 2^{0.8n}$, leading to an overall worst case runtime of $O^*(2^{0.4n})$.

In Jin and Wu's algorithm, Lemma 4 was used in developing the algorithm of Lemma 5 (the algorithm for the small $F(w_1, \dots, w_n)$ case), but was ignored in the case where $F(w_1, \dots, w_n)$ is large. In our improved algorithm, we show how the structural property of Lemma 4 can be exploited even when $F(w_1, \dots, w_n)$ is large. This will allow us to reduce the original Pigeonhole Equal Subset Sum instance to a small number of Equal Subset Sum instances, which have sizes much smaller than n (and still have large F-values).

3 Our Improvement for Pigeonhole Equal Subset Sum

In this section we prove our main Theorem 1. Let $w_1 < w_2 < \cdots < w_n$ be the input Pigeonhole Equal Subset Sum instance. Throughout, let $k \in \{0, 1, \ldots, n-1\}$ be such that

$$2^k \leq F(w_1, \ldots, w_n) < 2^{k+1}. \tag{3}$$

Our algorithm does not know the value of k in advance; instead, it tries all n possible values of k in parallel, and terminates as soon as any one of them succeeds. This only increases the runtime by an n factor. Hence, in the following we assume that the value of k is known.

We will separately consider the input items in $[n] \setminus [k] = \{k+1, \ldots, n\}$ and in $[k]$. At a high level, our algorithm performs the following two steps:

1. Guess a suitable pair of subsets $X, Y \subseteq [n] \setminus [k]$.
2. Find subsets $A_0, B_0 \subseteq [k]$ such that $w(A_0) - w(B_0) = w(Y) - w(X)$, i.e., $(A_0 \sqcup X, B_0 \sqcup Y)$ is a solution for Pigeonhole Equal Subset Sum.

For the first step, we will use Lemma 4 and Equation (3) to show that there are only poly(n) many pairs of $X, Y \subseteq [n] \setminus [k]$ that we need to consider. For the second step, we will reduce the task of finding A_0, B_0 to a smaller Equal Subset Sum instance with $(k+1)$ input integers and large F-value, which can be solved by Lemma 6. (In comparison, Jin and Wu's original algorithm applied Lemma 6 to solve Equal Subset Sum on n integers.)

Now we define the *close pairs*, which are the pairs of subsets (X, Y) that we need to consider for the first step:

▶ **Definition 7** (Close pairs). *Define the set of* close pairs *as*

$$\mathcal{C} := \{(X, Y) : X, Y \subseteq [n] \setminus [k] \text{ and } |w(X) - w(Y)| \leq (k+1)2^{k+1}\}.$$

Define the set of disjoint close pairs *as*

$$\mathcal{D} := \{(X, Y) \in \mathcal{C} : X \cap Y = \emptyset\}.$$

By the identity $w(X) - w(Y) = w(X \setminus Y) - w(Y \setminus X)$, we have

$$\mathcal{D} = \{(X \setminus Y, Y \setminus X) : (X, Y) \in \mathcal{C}\}. \tag{4}$$

The following simple lemma explains why it is sufficient to only consider close pairs:

▶ **Lemma 8.** *If $A, B \subseteq [n]$ and $w(A) = w(B)$, then $(A \setminus [k], B \setminus [k]) \in \mathcal{C}$.*

Proof. Since $w(A \setminus [k]) + w(A \cap [k]) = w(A) = w(B) = w(B \setminus [k]) + w(B \cap [k])$, we have

$$|w(A \setminus [k]) - w(B \setminus [k])| = |w(B \cap [k]) - w(A \cap [k])| \leq w([k]).$$

By the upper bounds in Lemma 4 and Equation (3), we have

$$w([k]) \leq \sum_{i=1}^{k} (2^{i-1} + F(w_1, \ldots, w_n)) < (k+1)2^{k+1}.$$

The claim then follows from the definition of $\mathcal{C}$. ◀

Now we bound the number of close pairs and disjoint close pairs:

▶ **Lemma 9.** *We have $|\mathcal{D}| \leq 200n^5$ and $|\mathcal{C}| \leq 200n^5 2^{n-k}$. Moreover, $\mathcal{D}$ can be computed in* poly(n) *time.*

Proof. First, we prove the claimed upper bound on $|\mathcal{D}|$. For $X \subseteq [n] \setminus [k]$, denote $\widetilde{w}(X) \coloneqq \sum_{i\in X} 2^{i-1}$. By Lemma 4 and Equation (3), for all $i \in [n]$,

$$|w_i - 2^{i-1}| \le i \cdot F(w_1, \ldots, w_n) \le n \cdot 2^{k+1}.$$

Hence, $|\widetilde{w}(X) - w(X)| \le n^2 \cdot 2^{k+1}$ for all $X \subseteq [n] \setminus [k]$. Therefore, for all $(X, Y) \in \mathcal{D}$, we have $|\widetilde{w}(X) - \widetilde{w}(Y)| \le |w(X) - w(Y)| + 2n^2 \cdot 2^{k+1} \le (k+1)2^{k+1} + 2n^2 \cdot 2^{k+1} \le 6n^2 2^k$. Now it remains to characterize all members of $\mathcal{D}'$ defined as

$$\mathcal{D}' \coloneqq \{(X, Y) \in 2^{[n]\setminus[k]} \times 2^{[n]\setminus[k]} : X \cap Y = \emptyset \text{ and } |\widetilde{w}(X) - \widetilde{w}(Y)| \le 6n^2 2^k\} \supseteq \mathcal{D}.$$

Note that $(\emptyset, \emptyset) \in \mathcal{D}'$. Now consider any $(X, Y) \in \mathcal{D}'$ not equal to $(\emptyset, \emptyset)$, and let i be the maximum element in $X \cup Y$. By symmetry, we assume $i \in X$ without loss of generality, and thus $Y \subseteq [i-1] \setminus [k]$. Then,

$$\begin{aligned}\widetilde{w}(X) - \widetilde{w}(Y) &= \Big(2^{i-1} + \sum_{j\in X\setminus\{i\}} 2^{j-1}\Big) - \Big(\sum_{j=k+1}^{i-1} 2^{j-1} - \sum_{j\in\{k+1,\ldots,i-1\}\setminus Y} 2^{j-1}\Big)\\ &= 2^k + \sum_{j\in X\setminus\{i\}} 2^{j-1} + \sum_{j\in\{k+1,\ldots,i-1\}\setminus Y} 2^{j-1} \ge 0.\end{aligned}$$

On the other hand, $\widetilde{w}(X) - \widetilde{w}(Y) \le 6n^2 2^k$ by definition of $\mathcal{D}'$. Hence, for every $j \in [i-1] \setminus [k]$ with $2^{j-1} \ge 6n^2 2^k$, we must have $j \in Y$ and $j \notin X$ (otherwise, 2^{j-1} would appear as a summand above, violating $\widetilde{w}(X) - \widetilde{w}(Y) \le 6n^2 2^k$). Then, to fully determine (X, Y), it only remains to decide for each integer $k+1 \le j < \log_2(6n^2 2^k) + 1$ whether j belongs to X, Y, or neither. There are at most $3^{\log_2(6n^2 2^k)+1-k} = 3^{\log_2(6n^2)+1} < 60n^4$ possible choices in total. Since there are up to n choices of i and two choices for whether i belongs to X or Y, we conclude that $|\mathcal{D}'| \le 1 + 2 \cdot n \cdot 60n^4 < 200n^5$.

The above argument can be easily converted into an algorithm that computes the set $\mathcal{D}'$ in $\text{poly}(n)$ time. Since $\mathcal{D}' \supseteq \mathcal{D}$, we have $|\mathcal{D}| \le 200n^5$ as claimed, and we can also compute $\mathcal{D}$ in $\text{poly}(n)$ time by checking every pair in $\mathcal{D}'$ for containment in $\mathcal{C}$.

Now we bound $|\mathcal{C}|$. Recall from Equation (4) that every $(X, Y) \in \mathcal{C} \subseteq 2^{[n]\setminus[k]} \times 2^{[n]\setminus[k]}$ can be mapped to $(X \setminus Y, Y \setminus X) \in \mathcal{D}$, and note that every $(X', Y') \in \mathcal{D}$ can only have at most 2^{n-k} pre-images in $\mathcal{C}$ under this mapping. Hence, $|\mathcal{C}| \le 2^{n-k}|\mathcal{D}| \le 200n^5 2^{n-k}$. ◀

The following lemma allows us to reduce the original problem to Equal Subset Sum instances on k or $k+1$ integers with large F-value:

▶ **Lemma 10.** *For sets $X, Y \subseteq [n] \setminus [k]$, define the k- or $(k+1)$-tuple $W_{X,Y}$ as*

$$W_{X,Y} \coloneqq \begin{cases} (w_1, \ldots, w_k) & \text{if } X = Y,\\ (w_1, \ldots, w_k, w(X) - w(Y)) & \text{if } X \ne Y.\end{cases} \tag{5}$$

Then, there must exist $(X, Y) \in \mathcal{D}$, such that $F(W_{X,Y}) \ge F(w_1, \ldots, w_n)/|\mathcal{C}|$.[3]

Before proving Lemma 10, we first use it to finish the proof of our main theorem:

[3] Note that in our definition of $W_{X,Y}$ for $(X, Y) \in \mathcal{D}$, the first case $X = Y$ happens only if $X = Y = \emptyset$. However we later will consider $W_{X,Y}$ where (X, Y) may not belong to $\mathcal{D}$.

Proof of Theorem 1 (assuming Lemma 10). As mentioned in the beginning of this section, we can assume the integer k defined by Equation (3) is known. We assume k satisfies

$$2^k \geq 200n^5 \cdot 2^{2n/3}, \tag{6}$$

since otherwise we can already use Lemma 5 to solve the Pigeonhole Equal Subset Sum instance $(w_1, \dots, w_n)$ in $O^*(\sqrt{2^{k+1}}) = O^*(2^{n/3})$ time as required.

Compute $\mathcal{D}$ with $|\mathcal{D}| \leq \mathrm{poly}(n)$ by Lemma 9 in $\mathrm{poly}(n)$ time. Enumerate each $(X, Y) \in \mathcal{D}$, and then solve the Equal Subset Sum instance $W_{X,Y}$ (defined in Equation (5)) using Lemma 6. If the algorithm in Lemma 6 successfully finds a solution $(U, V), U \neq V$ in the instance $W_{X,Y}$ (where we can assume $U \cap V = \emptyset$ without loss of generality), then we distinguish between two cases:

- **Case where $U, V \subseteq [k]$:**
  Then, since $w(U) = w(V)$ and $U \neq V$, we can return (U, V) as a solution for the original Pigeonhole Equal Subset Sum instance $w_1, \dots, w_n$.
- **Case where $k + 1 \in U \cup V$:**
  This can only happen in the $X \neq Y$ case, where $W_{X,Y}$ contains the extra $(k+1)$-th element $w(X) - w(Y)$. Without loss of generality assume $k + 1 \in U, k + 1 \notin V$. Then, define $A = (U \setminus \{k + 1\}) \sqcup X$ and $B = V \sqcup Y$, both of which are subsets of $[n]$. Note that $A \neq B$ due to $X \neq Y$. Then we have $w(A) = w(U \setminus \{k + 1\}) + w(X) = \big(w(V) - (w(X) - w(Y))\big) + w(X) = w(V) + w(Y) = w(B)$. So we can return (A, B) as a solution for the original Pigeonhole Equal Subset Sum instance $w_1, \dots, w_n$.

Now we analyze the runtime. By Lemma 10, for at least one $(X, Y) \in \mathcal{D}$, $F(W_{X,Y}) \geq F(w_1, \dots, w_n)/|\mathcal{C}| \geq (200n^5)^{-1} \cdot 2^{2k-n} =: \Delta$ (where we used Lemma 9 and Equation (3)). Then, applying Lemma 6 to solve $W_{X,Y}$ takes time $O^*((2^{2(k+1)}/\Delta)^{1/3}) = O^*\big(2^{n/3}\big)$. Note that the precondition $\Delta \geq 2^{(k+1)/2}$ required by Lemma 6 is satisfied due to Equation (6). We can try all $|\mathcal{D}| \leq \mathrm{poly}(n)$ possibilities for the pair (X, Y) in parallel and run the algorithm of Lemma 6 until the earliest of them succeeds. The total time complexity is still $O^*(2^{n/3})$. ◀

It remains to prove Lemma 10.

Proof of Lemma 10. We will show that there exists a pair $(X, Y) \in \mathcal{C}$ satisfying the desired bound $F(W_{X,Y}) \geq F(w_1, \dots, w_n)/|\mathcal{C}|$. Once this is shown, by $w(X) - w(Y) = w(X \setminus Y) - w(Y \setminus X)$ we have $W_{X\setminus Y, Y\setminus X} = W_{X,Y}$, so the pair $(X \setminus Y, Y \setminus X) \in \mathcal{D}$ also satisfies the desired bound, which would finish the proof.

We start by rewriting $F(W_{X,Y})$ as follows:

▷ Claim 11. For sets $X, Y \subseteq [n] \setminus [k]$ and integer t, define

$$h(X, t) := \big|\{S \subseteq [n] : S \setminus [k] = X \text{ and } w(S) = t\}\big|,$$

which is the number of ways to extend X into $[n]$ to achieve a sum of t, and define

$$g(X, Y, t) := \begin{cases} \max\{h(X, t) + h(Y, t) - 1, 0\} & \text{if } X \neq Y, \\ \max\{h(X, t) - 1, 0\} & \text{if } X = Y. \end{cases}$$

Then, $F(W_{X,Y}) = \sum_{t\in\mathbb{Z}} g(X, Y, t)$.

Proof of Claim 11. By definition of $h(X,t)$, observe that $h(X,t) = f_{t-w(X)}(w_1,\dots,w_k)$ (where $f_t(\cdot)$ was defined in Definition 3). Hence,

$$\begin{aligned} F(W_{X,X}) &= \sum_{t'\in\mathbb{Z}} \max\{f_{t'}(w_1,\dots,w_k) - 1, 0\} \\ &= \sum_{t\in\mathbb{Z}} \max\{f_{t-w(X)}(w_1,\dots,w_k) - 1, 0\} \\ &= \sum_{t\in\mathbb{Z}} \max\{h(X,t) - 1, 0\}. \end{aligned}$$

This finishes the proof for the $X = Y$ case. Now observe that

$$\begin{aligned} h(X,t) + h(Y,t) &= f_{t-w(X)}(w_1,\dots,w_k) + f_{t-w(Y)}(w_1,\dots,w_k) \\ &= f_{t-w(Y)}(w_1,\dots,w_k, w(X) - w(Y)), \end{aligned}$$

where the last equality can be seen from separately considering subsets of $\{w_1,\dots,w_k, w(X) - w(Y)\}$ which include $w(X) - w(Y)$ and those which do not. Therefore, if $X \neq Y$, then we similarly have

$$\begin{aligned} F(W_{X,Y}) &= \sum_{t'\in\mathbb{Z}} \max\{f_{t'}(w_1,\dots,w_k, w(X) - w(Y)) - 1, 0\} \\ &= \sum_{t\in\mathbb{Z}} \max\{h(X,t) + h(Y,t) - 1, 0\} \end{aligned}$$

as desired. ◀

Now we relate $F(w_1,\dots,w_n) = \sum_{t\in\mathbb{Z}} \max\{f_t(w_1,\dots,w_n) - 1, 0\}$ to the expressions from Claim 11. For any t with $f_t(w_1,\dots,w_n) \geq 1$, fix an arbitrary $A_t \subseteq [n]$ with $w(A_t) = t$, and let $X_t := A_t \setminus [k]$. Note that $h(X_t,t) \geq 1$. Then, by Lemma 8, every $Y \subseteq [n] \setminus [k]$ with $h(Y,t) \geq 1$ must satisfy $(X_t, Y) \in \mathcal{C}$. Hence,

$$\begin{aligned} f_t(w_1,\dots,w_n) - 1 &= -1 + \sum_{Y\subseteq[n]\setminus[k]:h(Y,t)\geq 1} h(Y,t) \qquad \text{(by definition of } f_t(\cdot) \text{ and } h(\cdot)\text{)} \\ &\leq -1 + \sum_{Y:(X_t,Y)\in\mathcal{C}} h(Y,t) \\ &= (h(X_t,t) - 1) + \sum_{Y:(X_t,Y)\in\mathcal{C}, Y\neq X_t} h(Y,t) \\ &\leq (h(X_t,t) - 1) + \sum_{Y:(X_t,Y)\in\mathcal{C}, Y\neq X_t} (h(X_t,t) + h(Y,t) - 1) \\ &\leq \sum_{Y:(X_t,Y)\in\mathcal{C}} g(X_t, Y, t) \\ &\leq \sum_{(X,Y)\in\mathcal{C}} g(X,Y,t). \end{aligned}$$

This means that for all $t \in \mathbb{Z}$,

$$\max\{f_t(w_1,\dots,w_n) - 1, 0\} \leq \sum_{(X,Y)\in\mathcal{C}} g(X,Y,t).$$

By summing over all $t \in \mathbb{Z}$ and substituting using Claim 11, we get

$$F(w_1,\dots,w_n) \leq \sum_{(X,Y)\in\mathcal{C}} F(W_{X,Y}).$$

Then, by averaging, there exists $(X, Y) \in \mathcal{C}$ such that $F(W_{X,Y}) \geq F(w_1, \ldots, w_n)/|\mathcal{C}|$ as claimed. ◀

4 A Polynomial-Space Algorithm

In this section we prove Theorem 2. We use the following two lemmas from Jin and Wu [15], which can be viewed as polynomial-space versions of Lemma 5 and Lemma 6 respectively:

▶ **Lemma 12** ([15, Lemma 11]). *Given parameter $1 \leq \Delta \leq 2^n/(3n^2)$, a Pigeonhole Equal Subset Sum instance $(w_1, \ldots, w_n)$ (where $w_i < w_{i+1}$) satisfying $F(w_1, \ldots, w_n) \leq \Delta$ and Equation (1) can be solved deterministically in $\mathrm{poly}(n)$ space and $O^*(\Delta)$ time.*

▶ **Lemma 13** ([15, Lemma 13]). *Given parameter $1 \leq \Delta \leq 2^n$, an Equal Subset Sum instance $(w_1, \ldots, w_n)$ with $F(w_1, \ldots, w_n) \geq \Delta$ can be solved in $O^*(2^{1.5n}/\Delta)$ time and $\mathrm{poly}(n)$ space by a randomized algorithm.*

Again, although [15, Lemma 13] was originally stated for the Pigeonhole Equal Subset Sum problem, it actually works for Equal Subset Sum as well because the proof does not use the pigeonhole promise.

Proof sketch of Theorem 2. We follow the same proof outline of Theorem 1, except that we replace the exponential-space subroutines, Lemma 5 and Lemma 6, by their polynomial-space counterparts, Lemma 12 and Lemma 13, respectively. The resulting algorithm is correct and runs in polynomial space. It only remains to re-analyze the time complexity.

Recall that we can assume the integer $k \in \{0, 1, \ldots, n-1\}$ satisfying $2^k \leq F(w_1, \ldots, w_n) < 2^{k+1}$ (Equation (3)) is known. Let $M \in [1, 2^n/(3n^2)]$ be determined later. If $2^k \leq M$, we choose to use Lemma 12 and solve the given Pigeonhole Equal Subset Sum instance in $O^*(2^{k+1}) = O^*(M)$ time. If $2^k > M$, we follow the main proof of Theorem 1 to reduce the original problem to $\mathrm{poly}(n)$ many Equal Subset Sum instances of size $k+1$ (or k) in which at least one instance has F-value $\Omega^*(2^{2k-n})$, so the total time complexity of applying Lemma 13 becomes $O^*(2^{1.5k}/2^{2k-n}) = O^*(2^{n-0.5k}) \leq O^*(2^n/\sqrt{M})$. By choosing $M = 2^{2n/3}$, the overall time complexity becomes $O^*(M + 2^n/\sqrt{M}) = O^*(2^{2n/3})$. ◀

5 Open Questions

We list a few open questions related to this work.

- Can we improve the $O^*(2^{n/3})$ time complexity for Pigeonhole Equal Subset Sum? Can we improve the $O^*(1.7088^n)$ time complexity for Equal Subset Sum [16]?
- Can we obtain fast *deterministic* algorithms for Pigeonhole Equal Subset Sum? Our algorithm (and Jin and Wu's algorithm [15]) uses Lemma 6 which is based on random subsampling. To the best of our knowledge, the $O^*(2^{n/2})$-time algorithms via meet-in-the-middle ([2] or [15, Section 2]) remain the fastest known deterministic algorithms.
- The authors of [2] proposed a more difficult *Modular version* of Pigeonhole Equal Subset Sum problem, and obtained a $O^*(2^{n/2})$-time algorithm. In this problem, we are given integers $w_1, \ldots, w_n$ and a modulus $m \leq 2^n - 1$, and need to find two distinct subsets $A, B \subseteq [n]$ such that $\sum_{i \in A} w_i \equiv \sum_{i \in B} w_i \pmod{m}$. Can we improve the $O^*(2^{n/2})$ time complexity for Modular Pigeonhole Equal Subset Sum?
- The k-SUM problem can be viewed as the fine-grained or parameterized version of the Subset Sum problem. Analogously, we propose the following Pigeonhole Equal k-SUM problem (for fixed integers $k \geq 2$):

PIGEONHOLE EQUAL k-SUM [19]
Input: k length-n arrays $A_1, \dots, A_k$ of integers from the range $[0, \frac{n^k-1}{k})$.
Output: two different k-tuples $(i_1, \dots, i_k), (j_1, \dots, j_k) \in [n]^k$ such that $A_1[i_1] + \cdots + A_k[i_k] = A_1[j_1] + \cdots + A_k[j_k]$.

This is a total search problem, and the brute force algorithm runs in $O(n^k)$ time. It is easy to improve it to $\widetilde{O}(n^{\lceil k/2 \rceil})$ time, using the standard binary search with meet-in-the-middle (following the same idea as the warm-up Pigeonhole Equal Subset Sum algorithm described in [15, Section 2], except that we replace the $O^*(2^{n/2})$-time subroutine for counting Subset Sum solutions by the analogous $\widetilde{O}(n^{\lceil k/2 \rceil})$-time algorithm for k-SUM). Can we improve the runtime to faster than $\widetilde{O}(n^{\lceil k/2 \rceil})$, say for the $k = 3$ case?

References

1 Shyan Akmal, Lijie Chen, Ce Jin, Malvika Raj, and R. Ryan Williams. Improved merlin-arthur protocols for central problems in fine-grained complexity. *Algorithmica*, 85(8):2395–2426, 2023. doi:10.1007/S00453-023-01102-6.

2 Jonathan Allcock, Yassine Hamoudi, Antoine Joux, Felix Klingelhöfer, and Miklos Santha. Classical and quantum algorithms for variants of subset-sum via dynamic programming. In *30th Annual European Symposium on Algorithms, ESA 2022, September 5-9, 2022, Berlin/Potsdam, Germany*, volume 244 of *LIPIcs*, pages 6:1–6:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. URL: https://arxiv.org/abs/2111.07059, doi:10.4230/LIPIcs.ESA.2022.6.

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

4 Per Austrin, Petteri Kaski, Mikko Koivisto, and Jesper Nederlof. Dense subset sum may be the hardest. In *Proceedings of the 33rd Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 47 of *LIPIcs*, pages 13:1–13:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.STACS.2016.13.

5 Per Austrin, Petteri Kaski, Mikko Koivisto, and Jesper Nederlof. Sharper upper bounds for unbalanced uniquely decodable code pairs. *IEEE Trans. Inf. Theory*, 64(2):1368–1373, 2018. doi:10.1109/TIT.2017.2688378.

6 Frank Ban, Kamal Jain, Christos H. Papadimitriou, Christos-Alexandros Psomas, and Aviad Rubinstein. Reductions in PPP. *Inf. Process. Lett.*, 145:48–52, 2019. doi:10.1016/j.ipl.2018.12.009.

7 Nikhil Bansal, Shashwat Garg, Jesper Nederlof, and Nikhil Vyas. Faster space-efficient algorithms for subset sum, k-sum, and related problems. *SIAM J. Comput.*, 47(5):1755–1777, 2018. doi:10.1137/17M1158203.

8 Anja Becker, Jean-Sébastien Coron, and Antoine Joux. Improved generic algorithms for hard knapsacks. In *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. doi:10.1007/978-3-642-20465-4_21.

9 Xavier Bonnetain, Rémi Bricout, André Schrottenloher, and Yixin Shen. Improved classical and quantum algorithms for subset-sum. In *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part II*, volume 12492 of *Lecture Notes in Computer Science*, pages 633–666. Springer, 2020. doi:10.1007/978-3-030-64834-3_22.

10 Lin Chen, Jiayi Lian, Yuchen Mao, and Guochuan Zhang. An improved pseudopolynomial time algorithm for subset sum. In *65th IEEE Annual Symposium on Foundations of Computer*

Science, FOCS 2024, Chicago, IL, USA, October 27-30, 2024, pages 2202–2216. IEEE, 2024. doi:10.1109/FOCS61266.2024.00129.

11 Xi Chen, Yaonan Jin, Tim Randolph, and Rocco A. Servedio. Average-case subset balancing problems. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 743–778. SIAM, 2022. doi:10.1137/1.9781611977073.33.

12 Xi Chen, Yaonan Jin, Tim Randolph, and Rocco A. Servedio. Subset sum in time $2^{\mathrm{n}/2}$ / poly(n). In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2023, September 11-13, 2023, Atlanta, Georgia, USA*, volume 275 of *LIPIcs*, pages 39:1–39:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.APPROX/RANDOM.2023.39.

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

14 Nick Howgrave-Graham and Antoine Joux. New generic algorithms for hard knapsacks. In *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. doi:10.1007/978-3-642-13190-5_12.

15 Ce Jin and Hongxun Wu. A faster algorithm for pigeonhole equal sums. In *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, July 8-12, 2024, Tallinn, Estonia*, volume 297 of *LIPIcs*, pages 94:1–94:11. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.ICALP.2024.94.

16 Marcin Mucha, Jesper Nederlof, Jakub Pawlewicz, and Karol Węgrzycki. Equal-subset-sum faster than the meet-in-the-middle. In *27th Annual European Symposium on Algorithms, ESA 2019, September 9-11, 2019, Munich/Garching, Germany*, volume 144 of *LIPIcs*, pages 73:1–73:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ESA.2019.73.

17 Jesper Nederlof. A short note on merlin-arthur protocols for subset sum. *Inf. Process. Lett.*, 118:15–16, 2017. doi:10.1016/j.ipl.2016.09.002.

18 Jesper Nederlof and Karol Węgrzycki. Improving Schroeppel and Shamir's algorithm for subset sum via orthogonal vectors. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 1670–1683. ACM, 2021. doi:10.1145/3406325.3451024.

19 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. doi:10.1016/S0022-0000(05)80063-7.

20 Tim Randolph. *Exact and Parameterized Algorithms for Subset Sum Problems*. PhD thesis, Columbia University, 2024. doi:10.7916/baym-5m55.

21 Katerina Sotiraki, Manolis Zampetakis, and Giorgos Zirdelis. PPP-completeness with connections to cryptography. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 148–158. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00023.

22 Henk C. A. van Tilborg. An upper bound for codes in a two-access binary erasure channel (corresp.). *IEEE Trans. Inf. Theory*, 24(1):112–116, 1978. doi:10.1109/TIT.1978.1055814.

23 Gerhard J. Woeginger. Open problems around exact algorithms. *Discret. Appl. Math.*, 156(3):397–405, 2008. doi:10.1016/J.DAM.2007.03.023.

24 Gerhard J. Woeginger and Zhongliang Yu. On the equal-subset-sum problem. *Inf. Process. Lett.*, 42(6):299–302, 1992. doi:10.1016/0020-0190(92)90226-L.

25 Stan Zhang. *Pigeonhole Equal Subset Sum in $O^*(2^{n/3})$*. Master's thesis, Massachusetts Institute of Technology, 2024. URL: https://hdl.handle.net/1721.1/156830.

A Proof of Lemma 4 by Jin and Wu

▶ **Lemma 4** ([15, Equation (8)]). *Suppose positive integers $w_1 < w_2 < \cdots < w_n$ satisfy $w([n]) < 2^n - 1$ and Equation (1). Then, for all $i \in [n]$,*

$$-i \cdot F(w_1, \ldots, w_n) \le w_i - 2^{i-1} \le F(w_1, \ldots, w_n). \tag{2}$$

The following proof of Lemma 4 was given by [15].

Proof. (Jin and Wu [15]) Recall our notations $w(A) = \sum_{i \in A} w_i$, $f_t = |\{A \subseteq [n] : w(A) = t\}|$. We abbreviate $F = F(w_1, \ldots, w_n) = \sum_{t \in \mathbb{Z}} \max\{f_t - 1, 0\}$. Recall our assumption from Equation (1):

$$w([i]) \ge 2^i - 1 \text{ for all } i \in [n-1]. \tag{1}$$

Since $w([n]) < 2^n - 1$, we know $f_t = 0$ for all $t \notin [0, 2^n - 1]$, and $\sum_{0 \le t < 2^n} f_t = 2^n$. Then,

$$F = \sum_{0 \le t < 2^n} \max\{f_t - 1, 0\} = \sum_{0 \le t < 2^n} (f_t - \mathbf{1}[f_t \ge 1]) = 2^n - \sum_{0 \le t < 2^n} \mathbf{1}[f_t \ge 1],$$

and thus

$$F = |\{0 \le t < 2^n : f_t = 0\}|. \tag{7}$$

Since $f_t = 0$ for all $w([n]) < t < 2^n$, from Equation (7) we know $F \ge 2^n - 1 - w([n])$, and hence $w([n]) \ge 2^n - 1 - F$. Combined with Equation (1) for $i \in [n-1]$, we get

$$w([i]) \ge 2^i - 1 - F \text{ for all } i \in [n]. \tag{8}$$

Now we have the following claim:

▷ Claim 14. For all $i \in [n]$,

$$w_i \le 2^{i-1} + F. \tag{9}$$

Summing Equation (9) over i gives

$$w([i]) \le 2^i - 1 + iF \tag{10}$$

for all $i \in [n]$.

Proof of Claim 14. Fix $i \in [n]$. Let M be the number of subsets $S \subseteq [n]$ with $w(S) < w_i$. Since $w_i < w_{i+1} < \cdots < w_n$, any such S must be contained in $[i-1]$, and thus $M \le 2^{i-1}$. On the other hand, $M = \sum_{t=0}^{w_i - 1} f_t \ge w_i - |\{0 \le t < w_i : f_t = 0\}| \ge w_i - F$ by Equation (7). Hence, $w_i \le M + F \le 2^{i-1} + F$. ◀

Comparing Equation (8) with Equation (10) gives

$$w_i = w([i]) - w([i-1]) \ge (2^i - 1 - F) - (2^{i-1} - 1 + (i-1)F) = 2^{i-1} - iF.$$

Together with Equation (9) we get the claimed bound

$$w_i - 2^{i-1} \in [-iF, F]$$

for all $i \in [n]$. ◀