A Faster Algorithm for Pigeonhole Equal Sums

Ce Jin
MIT, Cambridge, MA, USA

Hongxun Wu
University of California Berkeley, CA, USA

Abstract

An important area of research in exact algorithms is to solve Subset-Sum-type problems faster than meet-in-middle. In this paper we study *Pigeonhole Equal Sums*, a total search problem proposed by Papadimitriou (1994): given n positive integers $w_1, \dots, w_n$ of total sum $\sum_{i=1}^{n} w_i < 2^n - 1$, the task is to find two distinct subsets $A, B \subseteq [n]$ such that $\sum_{i \in A} w_i = \sum_{i \in B} w_i$.

Similar to the status of the Subset Sum problem, the best known algorithm for Pigeonhole Equal Sums runs in $O^*(2^{n/2})$ time, via either meet-in-middle or dynamic programming (Allcock, Hamoudi, Joux, Klingelhöfer, and Santha, 2022).

Our main result is an improved algorithm for Pigeonhole Equal Sums in $O^*(2^{0.4n})$ time. We also give a polynomial-space algorithm in $O^*(2^{0.75n})$ time. Unlike many previous works in this area, our approach does not use the representation method, but rather exploits a simple structural characterization of input instances with few solutions.

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

Keywords and phrases Subset Sum, Pigeonhole, PPP

Digital Object Identifier 10.4230/LIPIcs.ICALP.2024.94

Category Track A: Algorithms, Complexity and Games

Funding *Ce Jin*: Supported by NSF grants CCF-2129139 and CCF-2127597. Work done while visiting the Simons Institute for the Theory of Computing.
Hongxun Wu: Supported by Avishay Tal's Sloan Research Fellowship, NSF CAREER Award CCF-2145474, and Jelani Nelson's ONR grant N00014-18-1-2562.

Acknowledgements We thank Ryan Williams for useful discussions.

1 Introduction

The Subset Sum problem is an important NP-hard problem in computer science: given positive integers $w_1, w_2, \dots, w_n$ and a target integer t, find a subset $A \subseteq [n]$ such that $\sum_{i \in A} w_i = t$. Subset Sum can be solved in $O(2^{n/2})$ time by a simple meet-in-middle algorithm [14], and an important open problem is to improve it to $O(2^{(1/2-\varepsilon)n})$. A long line of research attempts to solve Subset Sum faster using the representation method [15] and connections to uniquely decodable code pairs [3, 4, 22], but these techniques have so far only succeeded on average-case inputs [15, 8, 9] or restricted classes of inputs [2, 3]. Nevertheless, significant progress has been made for other variants of Subset Sum, including Equal Sums [17], 2-Subset Sum and Shifted Sums [1] and more general subset balancing problems [12], as well as Subset Sum in other computational settings such as Merlin–Arthur protocols [18], low-space algorithms [6, 19], quantum algorithms [1], and algorithms with lower-order run time improvements [13]. The general hope is that the tools developed for solving these variant problems might one day help solve the original Subset Sum problem.

In this paper we study an interesting variant of Subset Sum called *Pigeonhole Equal Sums*:

> PIGEONHOLE EQUAL SUMS [20]
> **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 $\sum_{i \in A} w_i = \sum_{i \in B} w_i$.

Since there are 2^n subsets $S \subseteq [n]$ with only $2^n - 1$ possible subset sums $\sum_{i \in S} w_i \in \{0, 1, \ldots, 2^n - 2\}$ due to the promise, the pigeonhole principle guarantees that there exists a pair of subsets with the same subset sum.

Pigeonhole Equal Sums was introduced by Papadimitriou [20] as a natural example problem in the total search complexity class PPP. This problem has received attention in the TFNP literature [5, 21], and is conjectured to be PPP-complete [20].

From the algorithmic point of view, the current status of Pigeonhole Equal Sums is quite similar to that of the Subset Sum problem: a simple binary search with meet-in-middle solves Pigeonhole Equal Sums in $O^*(2^{n/2})$ time (see Section 2).[1] Allcock, Hamoudi, Joux, Klingelhöfer, and Santha [1, Theorem 6.2] gave another $O^*(2^{n/2})$-time algorithm based on dynamic programming (which is analogous to the alternative $O^*(2^{n/2})$-time Subset Sum algorithm from [3][2]). It remains open whether $O(2^{(1/2-\varepsilon)n})$ time is possible for Pigeonhole Equal Sums. Improvement of such type was achieved for the Equal Sums problem (without the pigeonhole promise) by Mucha, Nederlof, Pawlewicz, and Węgrzycki [17] via the representation method with $O(3^{(1/2-\varepsilon)n})$ run time for some $\varepsilon > 0.01$, but this result has no direct implications for Pigeonhole Equal Sums (for which the known $O^*(2^{n/2})$ time bound is already much better than $O(3^{n/2})$).

1.1 Our results

We give an algorithm that solves Pigeonhole Equal Sums faster than the previous $O^*(2^{n/2})$ running time [1].

▶ **Theorem 1** (Main). *Pigeonhole Equal Sums can be solved by a randomized algorithm in $O^*(2^{0.4n})$ time.*

Surprisingly, unlike previous works on other variants of Subset Sum, our algorithm does not use the representation method [15] or tools from coding theory [3, 4, 22]. Instead, our main insight is a simple structural characterization of Pigeonhole Equal Sums instances with few solutions.

Our techniques also yield a fast polynomial-space algorithm for Pigeonhole Equal Sums, in an analogous way to the previous $O(3^{(1-\varepsilon)n})$-time polynomial-space algorithm for Equal Sums [17].

▶ **Theorem 2.** *Pigeonhole Equal Sums can be solved by a randomized algorithm in $O^*(2^{0.75n})$ time and* poly(n) *space.*

For comparison, a straightforward algorithm based on binary search solves Pigeonhole Equal Sums in poly(n) space and $O^*(2^n)$ time (see the beginning of Section 4).

Theorem 1 and Theorem 2 will be proved in Section 3 and Section 4 respectively.

---

[1] We use $O^*(\cdot)$ to hide poly(n) factors.
[2] See also `https://youtu.be/cHimhXXIwcg?t=454`.

2 Preliminaries

Denote $[n] = \{1, \dots, n\}$. Let $O^*(\cdot), \Omega^*(\cdot)$ hide $\mathrm{poly}(n)$ factors, where n is the number of input integers in the Pigeonhole Equal Sums problem.

Denote $w(A) = \sum_{i \in A} w_i$ for $A \subseteq [n]$. The pigeonhole promise states $w([n]) < 2^n - 1$.

For a predicate p we define $\mathbf{1}[p] = 1$ if p is true and $\mathbf{1}[p] = 0$ if p is false.

We need the following well-known lemma.

▶ **Lemma 3** (Counting subset sums via meet-in-middle [14]). *Given integers $w_1, \dots, w_n$ and t, we can compute $\#\{S \subseteq [n] : w(S) \le t\}$ in $O^*(2^{n/2})$ time. Moreover, we can list $S \subseteq [n]$ such that $w(S) \le t$ in $O^*(1)$ additional time per S.*

Proof. Divide $[n]$ into $S_1 = \{1, \dots, \lfloor n/2 \rfloor\}$ and $S_2 = [n] \setminus S_1$, and every subset $S \subseteq [n]$ can be represented as $X \uplus Y, X \subseteq S_1, Y \subseteq S_2$. Compute and sort the two lists $A = \{w(X)\}_{X \subseteq S_1}$ and $B = \{w(Y)\}_{Y \subseteq S_2}$ of length $O(2^{n/2})$ each. Then for each $w(X) \in A$ we accumulate $|B \cap (-\infty, t - w(X)]|$ to the answer. It is easy to augment this algorithm to support listing. ◀

Pigeonhole Equal Sums via binary search

The following simple binary-search algorithm (described in [1, Remark 6.9 of arXiv version] and attributed to an anonymous referee) solves Pigeonhole Equal Sums in $O^*(2^{n/2})$ time: Maintain an interval $\{\ell, \ell+1, \dots, r\}$ (initialized to $\ell = 0, r = 2^n - 2$) that satisfies the pigeonhole invariant $r - \ell + 1 < \#\{S \subseteq [n] : \ell \le w(S) \le r\}$. Initially this invariant is satisfied due to $w([n]) \le 2^n - 2$. While $r > \ell$, pick the middle point $m = \lfloor \frac{\ell+r}{2} \rfloor$, and use meet-in-middle (Lemma 3) to compute $c_1 = \#\{S \subseteq [n] : \ell \le w(S) \le m\}$ and $c_2 = \#\{S \subseteq [n] : m+1 \le w(S) \le r\}$ in $O^*(2^{n/2})$ time. Then we shrink the interval to $\{\ell, \dots, m\}$ if $m - \ell + 1 < c_1$, or to $\{m+1, \dots, r\}$ if $r - m < c_2$ (the invariant guarantees that at least one holds). After $\lceil \log_2(2^n - 1) \rceil = n$ iterations we shrink to a singleton interval $\ell = r$. By the invariant, there exist two different $S_1, S_2 \subseteq [n]$ such that $w(S_1) = w(S_2) = \ell$, and we can report such S_1, S_2 using meet-in-middle (Lemma 3).

This binary-search strategy will be used in our improved algorithms as well.

3 The improved algorithm

Let the n input integers be sorted as $0 < w_1 < w_2 < \cdots < w_n$ (assuming no trivial solution $w_i = w_j$ exists).

An assumption on prefix sums

If any proper prefix $\{w_1, \dots, w_i\}$ $(i \le n-1)$ already satisfies the pigeonhole promise $w([i]) < 2^i - 1$, then we can instead solve the smaller Pigeonhole Equal Sums instance $\{w_1, \dots, w_i\}$ and obtain $A, B \subseteq [i], A \ne B$ with $w(A) = w(B)$. Hence, without loss of generality we assume such prefix does not exist, i.e.,

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

Frequencies f_t and parameter d

The *frequency* (also called bin size) of $t \in \mathbb{N}$ is the number of input subsets achieving sum t, denoted as $f_t = \#\{S \subseteq [n] : w(S) = t\}$. Since $w([n]) < 2^n - 1$, we know $f_t = 0$ for all $t \ge 2^n - 1$, and

$$\sum_{0 \le t < 2^n} f_t = 2^n. \tag{2}$$

Two different subsets achieving equal subset sum t imply $f_t > 1$. This motivates the following parameter,

$$d = \sum_{0 \le t < 2^n} \max\{0, f_t - 1\}, \tag{3}$$

which counts the (non-trivial) equality relations among all the 2^n subset sums. Using Equation (2), we can rewrite Equation (3) as $d = \sum_{0 \le t < 2^n} (f_t - \mathbf{1}[f_t \ge 1]) = 2^n - \sum_{0 \le t < 2^n} \mathbf{1}[f_t \ge 1]$, and thus obtain

$$d = \#\{0 \le t < 2^n : f_t = 0\}, \tag{4}$$

which counts the non-subset-sums in $\{0, 1, \dots, 2^n - 1\}$. In particular, $d < 2^n$.

The equivalence between Equation (3) and Equation (4) is powerful. In the following we will give two different algorithms for Pigeonhole Equal Sums. The first one works for small d by analyzing the structure of input instances with few non-subset-sums (by Equation (4)). The second one works when d is large and hence there are many solutions (by Equation (3)) which allow a subsampling approach. These two algorithms are summarized as follows:

▶ **Lemma 4.** *Given parameter $\Delta \le 2^n/(3n^2)$, Pigeonhole Equal Sums with $d \le \Delta$ can be solved deterministically in $O^*(\sqrt{\Delta})$ time.*

▶ **Lemma 5.** *Given parameter $2^{n/2} \le \Delta < 2^n$, Pigeonhole Equal Sums with $d \ge \Delta$ can be solved in $O^*((2^{2n}/\Delta)^{1/3})$ time by a randomized algorithm.*

Combining these two lemmas implies our main result:

Proof of Theorem 1. Set $\Delta = 2^{0.8n}$ so that the two time bounds in Lemma 4 and Lemma 5 are balanced to $O^*(2^{0.4n})$. Given an instance of Pigeonhole Equal Sums (with unknown d), we run both algorithms in parallel, and return the answer of whichever terminates first. ◀

3.1 Small d case via structural characterization

In this section we prove Lemma 4. Assume $d \le \Delta \le 2^n/(3n^2)$ and Δ is known.

Since $f_t = 0$ for all $w([n]) < t < 2^n$, from Equation (4) we know $d \ge 2^n - 1 - w([n])$, and hence $w([n]) \ge 2^n - 1 - d \ge 2^n - 1 - \Delta$. Combined with Equation (1) for $i \in [n-1]$, we get the following lower bound

$$w([i]) \ge 2^i - 1 - \Delta \text{ for all } i \in [n]. \tag{5}$$

The key step is to complement Equation (5) with a nearly matching upper bound:

▶ **Lemma 6.** *For all $i \in [n]$,*

$$w_i \le 2^{i-1} + \Delta. \tag{6}$$

Summing Equation (6) over i gives

$$w([i]) \le 2^i - 1 + i\Delta \tag{7}$$

for all $i \in [n]$.

Proof. 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, $M = \sum_{t=0}^{w_i - 1} f_t \ge w_i - \#\{0 \le t < w_i : f_t = 0\} \ge w_i - d$ by Equation (4). Hence, $w_i \le M + d \le 2^{i-1} + \Delta$. ◀

Comparing Equation (5) with Equation (7) gives the lower bound

$$w_i = w([i]) - w([i-1]) \geq (2^i - 1 - \Delta) - (2^{i-1} - 1 + (i-1)\Delta) = 2^{i-1} - i\Delta,$$

which is very close to the upper bound from Equation (6). Together we get

$$w_i - 2^{i-1} \in [-i\Delta, \Delta] \tag{8}$$

for all $i \in [n]$.

Equation (8) gives a very rigid structure of the large input numbers. In the next lemma we exploit this structure to improve the naive meet-in-middle subset sum counting algorithm from Lemma 3.

▶ **Lemma 7.** *For any given $T < 2^n$, we can compute $\sum_{t=0}^{T} f_t$ in $O^*(\sqrt{\Delta})$ time.*

Proof. Let i^* be the minimum $i^* \in [n]$ such that $2^{i^*} \geq 3n^2\Delta$, which exists by our assumption $\Delta \leq 2^n/(3n^2)$. Let $A = \{1, 2, \ldots, i^*\}$ and $B = \{i^*+1, \ldots, n\}$.

By Equation (7), $w(A) < 2^{i^*} + n\Delta$.

For every $B' \subseteq B$, by Equation (8) we have

$$\left|w(B') - \sum_{j \in B'} 2^{j-1}\right| \leq \sum_{j \in B'} |w_j - 2^{j-1}| \leq \sum_{j \in B'} j\Delta \leq n^2\Delta.$$

In other words, the subset sums of $\{w_j\}_{j \in B}$ are $n^2\Delta$-additively approximated by the subset sums of $\{2^{j-1}\}_{j \in B}$. The subset sums of the latter set form an arithmetic progression $\{k \cdot 2^{i^*} : 0 \leq k < 2^{n-i^*}\}$, namely all n-bit binary numbers whose lowest i^* bits are zeros. Notably, this arithmetic progression is very sparse: its difference 2^{i^*} is large enough compared to $w(A) < 2^{i^*} + n\Delta$.

Given query T, we want to count the number of pairs (A', B') $(A' \subseteq A, B' \subseteq B)$ such that $w(A') + w(B') \leq T$. To do this, we enumerate $B' \subseteq B$, and consider three cases (the non-trivial case is Case 3, where $w(B')$ and $\sum_{j \in B'} 2^{j-1}$ are close to T):

- **Case 1:** $\sum_{j \in B'} 2^{j-1} \leq T - 2^{i^*} - (n+n^2)\Delta$.
  Then, for all $A' \subseteq A$, we have $w(A') + w(B') \leq w(A) + w(B') \leq (2^{i^*} + n\Delta) + (n^2\Delta + \sum_{j \in B'} 2^{j-1}) \leq T$. Hence B' contributes $2^{|A|}$ many pairs (A', B').
- **Case 2:** $\sum_{j \in B'} 2^{j-1} > T + n^2\Delta$.
  Then, for all $A' \subseteq A$, we have $w(A') + w(B') \geq w(B') \geq \sum_{j \in B'} 2^{j-1} - n^2\Delta > T$. Hence B' does not contribute any pairs (A', B').
- **Case 3:** otherwise, $\sum_{j \in B'} 2^{j-1} \in (T - 2^{i^*} - (n+n^2)\Delta, T + n^2\Delta]$.
  This interval has length $2^{i^*} + (n+n^2)\Delta + n^2\Delta \leq 2 \cdot 2^{i^*}$ by our choice of i^*. Since $\sum_{j \in B'} 2^{j-1}$ is a multiple of 2^{i^*} in this interval, it has at most two possibilities, namely $2^{i^*} \cdot \lfloor \frac{T-(n+n^2)\Delta}{2^{i^*}} \rfloor$ and $2^{i^*} \cdot \left(\lfloor \frac{T-(n+n^2)\Delta}{2^{i^*}} \rfloor + 1\right)$, and then B' is uniquely determined by the binary decomposition of $\sum_{j \in B'} 2^{j-1}$. For each possible B', we count the number of $A' \subseteq A$ such that $w(A') \leq T - w(B')$ using meet-in-middle (Lemma 3) with time complexity $O^*(2^{|A|/2}) = O^*(2^{i^*/2}) = O^*(\sqrt{\Delta})$ by the definition of i^*.

Note that in $O^*(1)$ time we can easily find the (at most two) subsets B' satisfying Case 3, and also count the total contribution of Case 1. Hence the overall time complexity is $O^*(\sqrt{\Delta})$. ◀

Using Lemma 7 we can solve Pigeonhole Equal Sums using binary search, in the same way as described in the last paragraph of Section 2. The running time is $O^*(\sqrt{\Delta})$. This finishes the proof of Lemma 4.

3.2 Large d case via subsampling

In this section we prove Lemma 5. Assume $2^{n/2} \leq \Delta \leq d < 2^n$, and Δ is known. We first use $d = \sum_{0 \leq t < 2^n} \max\{0, f_t - 1\}$ (Equation (3)) to show that many subset sums t have large f_t, which then allows us to use subsampling to speed up the modular dynamic programming approach of [1, 3].

▶ **Lemma 8.** *There exists a $j \in \{0, 1, \dots, n-1\}$ such that $\#\{t : f_t > 2^j\} > \frac{\Delta}{2^{j+1}n}$.*

Proof. By definition of d in Equation (3),

$$\Delta \leq d = \sum_{t: f_t > 1} (f_t - 1) \leq \sum_{0 \leq j < n} \#\{t : 2^j < f_t \leq 2^{j+1}\} \cdot (2^{j+1} - 1). \tag{9}$$

If the claimed inequality fails for all j, then

$$[\text{RHS of Equation (9)}] \leq \sum_{0 \leq j < n} \frac{\Delta}{2^{j+1}n} \cdot (2^{j+1} - 1) < \Delta,$$

a contradiction. ◀

Our algorithm enumerates all $j \in \{0, 1, \dots, n-1\}$ (increasing the time complexity by a factor of $n = O^*(1)$), and from now on we assume j satisfies the inequality in Lemma 8. Define

$$h := 2^j + 1 \geq 2, \quad m := \left\lceil \frac{\Delta}{2^{j+1}n} \right\rceil > \frac{\Delta}{2hn}, \quad \text{and } X := \{t \in [2^n] : f_t \geq h\}. \tag{10}$$

Here we defined the set X of frequent subset sums only for the sake of analysis. By Lemma 8,

$$|X| \geq m. \tag{11}$$

Readers are encouraged to focus on the case of $h = 2$ and $m \geq \Omega^*(\Delta)$ (which is the hardest case for our algorithm) at first read.

We first describe the behavior of our algorithm: Let $p \in [P, 2P]$ be a uniformly random prime (for some parameter $2 \leq P \leq 2m$ to be determined later in the "Time complexity" paragraph). For each $r \in \mathbb{Z}_p$, define bin $B_r := \{S \subseteq [n] : w(S) \equiv r \pmod p\}$. The algorithm picks a random bin index $r^* \in \mathbb{Z}_p$, and subsamples $C \subseteq B_{r^*}$ by keeping each $S \in B_{r^*}$ with probability α independently (for some $0 < \alpha \leq \frac{1}{2h}$ to be determined later in the "Success probability" paragraph). Finally, a pair of distinct $S, S' \in C$ with $w(S) = w(S')$ is reported (if exists).

Now we explain how to implement the algorithm above via dynamic programming (DP) similarly to [1, 3]. Build the DP table $D_{i,r} = \#\{S \subseteq [i] : w(S) \equiv r \pmod p\}$ (where $0 \leq i \leq n$ and $r \in \mathbb{Z}_p$) in $O^*(p)$ overall time via the transition $D_{i,r} = D_{i-1,r} + D_{i-1,(r-w_i) \bmod p}$ with initial values $D_{0,r} = \mathbf{1}[r = 0]$. This DP computes the size of every bin $|B_r| = D_{n,r}$. Furthermore, for any bin B_r and integer $k \in [|B_r|]$, we can report the rank-k set S in B_r (in lexicographical order, where larger indices are compared first) by backtracing in the DP table in $O^*(1)$ time. Then, in order to subsample a collection of sets $C \subseteq B_{r^*}$ at rate α, we can first subsample their ranks in $[|B_{r^*}|]$ (in near-linear time in the output size, see e.g., [10]), and then recover the actual sets by backtracing.

Success probability

We study how the frequent subset sums, $X = \{t : f_t \geq h\}$, are distributed to the bins modulo a random prime p, using an argument similar to [3]. Setting

$$k := \lceil \frac{m}{4P} \rceil, \tag{12}$$

the following lemma shows that the bin B_{r^*} receives at least k frequent subset sums, with $\Omega^*(1)$ probability.

▶ **Lemma 9.** *With at least $\Omega(1/n)$ probability over the choice of prime $p \in [P, 2P]$ and $r^* \in \mathbb{Z}_p$, there are at least k integers $t \in \mathbb{N}$ such that $\#\{S \in B_{r^*} : w(S) = t\} \geq h$.*

Proof. Since $|X| \geq m$ by Equation (11), we arbitrarily pick $X' \subseteq X$ with $|X'| = m$ for the sake of analysis. Let $c_{r,p} := \{t \in X' : t \equiv r \pmod p\}$. Then,

$$\begin{aligned}
\mathop{\mathbf{E}}_{p\in[P,2P]}\Big[\sum_{r\in\mathbb{Z}_p} c_{r,p}^2\Big] &= \sum_{x\in X', y\in X'} \mathop{\mathbf{Pr}}_{p\in[P,2P]}[p \mid x-y] \\
&\leq m + m^2 \cdot \frac{\log_P 2^n}{\Omega(P/\ln P)} && \text{(by } |x-y| \leq 2^n \text{ and the density of primes)} \\
&\leq O(n \cdot m^2/P). && \text{(by the assumption that } P \leq 2m)
\end{aligned}$$

Then by Markov's inequality, with 0.9 success probability over the choice of p, we have $\sum_{r\in\mathbb{Z}_p} c_{r,p}^2 \leq O(n \cdot m^2/P)$. Conditioned on this happening, by Cauchy–Schwarz inequality we have

$$\begin{aligned}
\sum_{r\in\mathbb{Z}_p} \mathbf{1}[c_{r,p} \geq \tfrac{m}{2p}] &\geq \frac{\left(\sum_{r\in\mathbb{Z}_p} \mathbf{1}[c_{r,p} \geq \frac{m}{2p}] \cdot c_{r,p}\right)^2}{\sum_{r\in\mathbb{Z}_p} c_{r,p}^2} \\
&\geq \frac{\left((\sum_{r\in\mathbb{Z}_p} c_{r,p}) - p \cdot \frac{m}{2p}\right)^2}{O(n \cdot m^2/P)} = \frac{(|X'| - m/2)^2}{O(n \cdot m^2/P)} = \frac{(m/2)^2}{O(n \cdot m^2/P)} = \Omega(P/n),
\end{aligned}$$

and hence, by our choice of $k = \lceil \frac{m}{4P} \rceil \leq \lceil \frac{m}{2p} \rceil$,

$$\mathop{\mathbf{Pr}}_{r^*\in\mathbb{Z}_p}[c_{r^*,p} \geq k] \geq \mathop{\mathbf{Pr}}_{r^*\in\mathbb{Z}_p}[c_{r^*,p} \geq \tfrac{m}{2p}] \geq \frac{\Omega(P/n)}{p} = \Omega(1/n).$$

Conditioned on $c_{r^*,p} \geq k$ happening, we have at least k integers $t \in X' \subseteq X$ such that $t \equiv r^* \pmod p$. By definitions of B_{r^*} and X, this implies that there are at least k integers $t \in \mathbb{N}$ such that $\#\{S \in B_{r^*} : w(S) = t\} \geq h$, with overall success probability at least $0.9 \cdot \Omega(1/n) = \Omega(1/n)$ over the choice of p and r^*. ◀

Recall our algorithm subsamples $C \subseteq B_{r^*}$ at rate $\alpha \in (0, \frac{1}{2h}]$, and fails iff $w(S)$ are distinct for all $S \in C$. The failure probability of this step can be derived from the following lemma:

▶ **Lemma 10.** *Let B' be a collection of kh colored balls ($h \geq 2, k \geq 1$), with exactly h balls of color i for each color $i \in [k]$. Let $C' \subseteq B'$ be an i.i.d. subsample at rate $\alpha \in [0, \frac{1}{2h}]$. Then C' contains distinct colors with at most $\exp(-kh(h-1)\alpha^2/4)$ probability.*

Proof. For each color $i \in [k]$, by Bernoulli's inequality, the probability that C' includes exactly two balls of color i is $\binom{h}{2}\alpha^2(1-\alpha)^{h-2} \geq \binom{h}{2}\alpha^2\big(1-(h-2)\alpha\big) \geq \binom{h}{2}\alpha^2/2$. Hence, the probability that C' includes at most one ball of every color $i \in [k]$ is at most $\big(1-\binom{h}{2}\alpha^2/2\big)^k \leq \exp\big(-k\binom{h}{2}\alpha^2/2\big) = \exp(-kh(h-1)\alpha^2/4)$. ◀

We think of each set $S \in B_{r^*}$ as a ball of color $w(S)$, and apply Lemma 10 to the k integers (colors) $t \in \mathbb{N}$ ensured by Lemma 9, each having at least h sets (balls) $S \in B_{r^*}$ with $w(S) = t$. We set the sample rate to be

$$\alpha := \frac{1}{2h\sqrt{k}} \leq \frac{1}{2h}. \tag{13}$$

Then the failure probability of the subsampling step is at most

$$\exp(-kh(h-1)\alpha^2/4) = \exp(-\tfrac{h-1}{16h}) \leq \exp(-1/32).$$

Overall, the probability that the algorithm successfully finds a solution is at least $\Omega(1/n) \cdot (1 - \exp(-1/32)) \geq \Omega(n^{-1})$.

Time complexity

The mod-p DP runs in $O^*(p) \leq O^*(P)$ time. Since the bins have total size $\sum_{r \in \mathbb{Z}_p} |B_r| = 2^n$, the chosen bin B_{r^*} has expected size $\mathbf{E}_{r^* \in \mathbb{Z}_p}[|B_{r^*}|] = 2^n/p \leq 2^n/P$, and hence the subsample $C \subseteq B_{r^*}$ has expected size $\mathbf{E}[|C|] \leq \alpha 2^n/P$. To detect a solution $S, S' \in C$ with $w(S) = w(S')$, we simply sort C in near-linear time. Hence the total expected running time is $O^*(P + \alpha 2^n/P)$. By Markov's inequality, with probability at least $1 - n^{-10}$, the algorithm terminates in $O^*(P + \alpha 2^n/P)$ time. By a union bound, the algorithm successfully finds a solution in time $O^*(P + \alpha 2^n/P)$ with probability at least $\Omega(n^{-1}) - n^{-10} \geq \Omega(n^{-1})$. This success probability can be boosted to 0.99 by repeating the algorithm $O(n)$ times.

Recall from Equations (12) and (13) that $\alpha = \frac{1}{2h\sqrt{k}} = \frac{1}{2h\sqrt{\lceil m/4P \rceil}} \leq \frac{\sqrt{P}}{h\sqrt{m}}$, so the run time is (ignoring poly(n) factors)

$$P + \alpha 2^n/P \leq P + \frac{2^n}{h\sqrt{mP}}.$$

Recall $h = 2^j + 1$ (where $0 \leq j \leq n-1$) and $m = \lceil \frac{\Delta}{2^{j+1}n} \rceil$, and hence $hm < h(1 + \frac{\Delta}{2^{j+1}n}) \leq h + \frac{\Delta}{n} < (2^{n-1}+1) + \frac{2^n}{n} \leq 2^n$ (assuming $n \geq 3$). Now we set

$$P := 2m \cdot \min\left\{1, (\frac{2^n}{hm^2})^{2/3}\right\},$$

and we first need to verify the requirement $2 \leq P \leq 2m$ introduced earlier: The upper bound is obvious. To see the lower bound, note that $2m \geq 2$ and $2m \cdot \left(\frac{2^n}{hm^2}\right)^{2/3} = 2\left(\frac{2^{2n}}{h^2 m}\right)^{1/3} \geq 2\left(\frac{2^{2n}}{(hm)^2}\right)^{1/3} \geq 2$ (using the inequality $hm \leq 2^n$ we just showed).

Hence, the overall running time is at most (ignoring poly(n) factors)

$$\begin{aligned}
P + \frac{2^n}{h\sqrt{mP}} &\leq 2m\left(\frac{2^n}{hm^2}\right)^{2/3} + \frac{2^n}{h\sqrt{m \cdot 2m}} \cdot \max\left\{1, \left(\frac{hm^2}{2^n}\right)^{1/3}\right\} \\
&= 2 \cdot \frac{2^{2n/3}}{h^{2/3}m^{1/3}} + \frac{1}{\sqrt{2}} \max\left\{\frac{2^n}{hm}, \frac{2^{2n/3}}{h^{2/3}m^{1/3}}\right\} \\
&\leq O\left(\frac{2^{2n/3}}{h^{2/3}m^{1/3}} + \frac{2^n}{hm}\right) \\
&\leq O^*\left(\frac{2^{2n/3}}{h^{1/3}\Delta^{1/3}} + \frac{2^n}{\Delta}\right) && \text{(by } hm > \tfrac{\Delta}{2n} \text{ from Equation (10))} \\
&\leq O^*\left(\frac{2^{2n/3}}{\Delta^{1/3}}\right). && \text{(by } h > 1 \text{ and the assumption that } \Delta \geq 2^{n/2})
\end{aligned}$$

This finishes the proof of Lemma 5.

4 A polynomial-space algorithm

We now consider poly(n)-space algorithms for Pigeonhole Equal Sums. The straightforward binary search approach (described at the end of Section 2) can be adapted to run in $O^*(2^n)$ time and poly(n) space: instead of using meet-in-middle (Lemma 3, which requires large space), we count the number of valid subsets $S \subseteq [n]$ by brute force in $O^*(2^n)$ time and only poly(n) space.

We improve this $O^*(2^n)$ running time using the ideas from earlier sections. Again, consider two cases depending on whether parameter d from Equation (3) is small or large.

▶ **Lemma 11.** *Given parameter $\Delta \leq 2^n/(3n^2)$, Pigeonhole Equal Sums with $d \leq \Delta$ can be solved deterministically in* poly(n) *space and $O^*(\Delta)$ time.*

Proof Sketch. The proof is almost the same as Lemma 4 (see Section 3.1), with the only difference in Case 3 from the proof of Lemma 7: instead of using meet-in-middle, here we count the valid subsets $A' \subseteq A$ by brute force in $O^*(2^{|A|}) = O^*(2^{i^*}) = O^*(\Delta)$ time and only poly(n) space. ◀

To solve the large d case, we need the low-space element distinctness algorithm by Beame, Clifford, and Machmouchi [7] (generalized in [6], and with a non-standard assumption removed by [11, 16]). This algorithm was also previously used for Subset Sum [6] and Equal Sums [17]. The following statement can be inferred from [11, Section 4.2 (proof of Theorem 1.1)].

▶ **Theorem 12** (Low-space Element Distinctness, [7, 6, 11]). *Given random access to an integer list $a_1, \dots, a_N$ (where $a_i \in [\text{poly}(N)]$) that contains at least one pair $(i, j) \in [N] \times [N]$ with $a_i = a_j, i \neq j$, there is a randomized algorithm that reports such a pair using* $\text{poly}\log N$ *working memory and*

$$O\left(\frac{N\sqrt{F_2}}{F_2 - N} \cdot \text{poly}\log N\right)$$

time, where $F_2 = \sum_{i=1}^{N} \sum_{j=1}^{N} \mathbf{1}[a_i = a_j] \in [N+2, N^2]$.[3]

▶ **Lemma 13.** *Given parameter $1 \leq \Delta \leq 2^n$, Pigeonhole Equal Sums with $d \geq \Delta$ can be solved in $O^*(2^{1.5n}/\Delta)$ time and* poly(n) *space by a randomized algorithm.*

Proof. Apply Theorem 12 to the list $\{w(A)\}_{A\subseteq[n]}$ of length $N = 2^n$ and we obtain a pair of distinct $A, A' \subseteq [n]$ with $w(A) = w(A')$ as desired. The space complexity is $\text{poly}\log(2^n) = \text{poly}(n)$. To analyze the time complexity, note that

$$F_2 - 2^n = \sum_{A\subseteq[n]} \sum_{\substack{B\subseteq[n]\\B\neq A}} \mathbf{1}[w(A) = w(B)] = \sum_{0\leq t<2^n} f_t(f_t - 1) \geq \sum_{0\leq t<2^n} \max\{0, f_t - 1\} \overset{\text{Eq. (3)}}{=} d \geq \Delta,$$

so the time bound is (ignoring poly(n) factors)

$$\frac{2^n\sqrt{F_2}}{F_2 - 2^n} < \frac{2^{0.5n}F_2}{F_2 - 2^n} = 2^{0.5n}\left(1 + \frac{2^n}{F_2 - 2^n}\right) \leq 2^{0.5n}\left(1 + \frac{2^n}{\Delta}\right) \leq \frac{2 \cdot 2^{1.5n}}{\Delta}$$

as claimed. ◀

---

[3] We have $F_2 \geq N + 2$ due to the following $(N + 2)$ pairs: $(1, 1), (2, 2), \dots, (N, N)$ and $(i, j), (j, i)$, where $a_i = a_j$ $(i \neq j)$.

Combining the two lemmas gives the desired result.

Proof of Theorem 2. Set $\Delta = 2^{0.75n}$ so that the two time bounds in Lemma 11 and Lemma 13 are balanced to $O^*(2^{0.75n})$. Given an instance of Pigeonhole Equal Sums (with unknown d), we run both algorithms in parallel, and return the answer of whichever terminates first. ◀

5 Open problems

Allcock et al. [1] proposed a modular variant of the Pigeonhole Equal Sums problem: given integers $w_1, \ldots, w_n$ and a modulus $m \leq 2^n - 1$, find two distinct subsets $A, B \subseteq [n]$ such that $\sum_{i \in A} w_i \equiv \sum_{i \in B} w_i \pmod{m}$. They obtained a $O^*(2^{n/2})$-time algorithm for this problem. Can this result be improved as well?

Can we obtain faster algorithms for other problems in PPP (e.g., [5, 21])?

References

1 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. doi:10.4230/LIPIcs.ESA.2022.6.

2 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.

3 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.

4 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.

5 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.

6 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.

7 Paul Beame, Raphaël Clifford, and Widad Machmouchi. Element distinctness, frequency moments, and sliding windows. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 290–299, 2013. doi:10.1109/FOCS.2013.39.

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 Karl Bringmann and Konstantinos Panagiotou. Efficient sampling methods for discrete distributions. *Algorithmica*, 79(2):484–508, 2017. doi:10.1007/S00453-016-0205-0.

11 Lijie Chen, Ce Jin, R. Ryan Williams, and Hongxun Wu. Truly low-space element distinctness and subset sum via pseudorandom hash functions. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1661–1678, 2022. doi:10.1137/1.9781611977073.67.

12 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.

13 Xi Chen, Yaonan Jin, Tim Randolph, and Rocco A. Servedio. Subset sum in time $2^{n/2}$ / 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.

14 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.

15 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.

16 Xin Lyu and Weihao Zhu. Time-space tradeoffs for element distinctness and set intersection via pseudorandomness. In *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 5243–5281. SIAM, 2023. doi:10.1137/1.9781611977554.ch190.

17 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.

18 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.

19 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.

20 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.

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.