Classical and Quantum Algorithms for Variants of Subset-Sum via Dynamic Programming

Jonathan Allcock* Yassine Hamoudi† Antoine Joux‡

Felix Klingelhöfer§ Miklos Santha¶

July 22, 2022

Abstract

Subset-Sum is an NP-complete problem where one must decide if a multiset of n integers contains a subset whose elements sum to a target value m. The best-known classical and quantum algorithms run in time $\widetilde{O}(2^{n/2})$ and $\widetilde{O}(2^{n/3})$, respectively, based on the well-known meet-in-the-middle technique. Here we introduce a novel classical dynamic-programming-based data structure with applications to Subset-Sum and a number of variants, including Equal-Sums (where one seeks two disjoint subsets with the same sum), 2-Subset-Sum (a relaxed version of Subset-Sum where each item in the input set can be used twice in the summation), and Shifted-Sums, a generalization of both of these variants, where one seeks two disjoint subsets whose sums differ by some specified value.

Given any modulus p, our data structure can be constructed in time $O(n^2 p)$, after which queries can be made in time $O(n^2)$ to the lists of subsets summing to any value modulo p. We use this data structure in combination with variable-time amplitude amplification and a new quantum pair finding algorithm, extending the quantum claw finding algorithm to the multiple solutions case, to give an $O(2^{0.504n})$ quantum algorithm for Shifted-Sums. This provides a notable improvement over the best-known $O(2^{0.773n})$ classical running time established recently by Mucha et al. [MNPW19]. Incidentally, we obtain new $\widetilde{O}(2^{n/2})$ and $\widetilde{O}(2^{n/3})$ classical and quantum algorithms for Subset-Sum, not based on the seminal meet-in-the-middle approach of Horowitz and Sahni [HS74]. We also study Pigeonhole Equal-Sums and Pigeonhole Modular Equal-Sums, variants of Equal-Sums where the existence of a solution is guaranteed by the pigeonhole principle. For the former problem, we give faster classical and quantum algorithms with running time $\widetilde{O}(2^{n/2})$ and $\widetilde{O}(2^{2n/5})$, respectively. For the more general modular problem, we give a classical algorithm that also runs in time $\widetilde{O}(2^{n/2})$.

*Tencent Quantum Laboratory, Hong Kong. jonallcock@tencent.com

†Simons Institute for the Theory of Computing, University of California, Berkeley. ys.hamoudi@gmail.com

‡CISPA Helmholtz Center for Information Security. joux@cispa.de

§G-SCOP, Université Grenoble Alpes. felix.klingelhofer@grenoble-inp.fr

¶Centre for Quantum Technologies and MajuLab, National University of Singapore. miklos.santha@gmail.com

1 Introduction

Subset-Sum is the problem of deciding whether a given multiset of n integers has a subset whose elements sum to a target integer m.

Problem 1 (Subset-Sum)**.** Given a multiset $\{a_1, \ldots, a_n\}$ of positive integers and a target integer m, find a subset $S \subseteq [n]$ such that $\sum_{i \in S} a_i = m$.

It is often useful to express Subset-Sum using inner product notation. We set $\bar{a} = (a_1, \ldots, a_n) \in \mathbb{N}^n$, where the elements are taken in arbitrary order, and the task is to find $\bar{e} \in \{0,1\}^n$ such that $\bar{a} \cdot \bar{e} = \sum_{i=1}^n a_i e_i = m$. The problem is famously NP-complete and featured on Karp's list of 21 NP-complete problems [Kar72] in 1972 (under the name of knapsack). It can be solved classically in time $\widetilde{O}(2^{n/2})$ via the *meet-in-the-middle* technique [HS74]. Whether this problem can be solved in time $\widetilde{O}(2^{(1/2-\delta)n})$, for some $\delta > 0$, is an important open question, but we know that the Exponential Time Hypothesis implies that Subset-Sum cannot be computed in time $m^{o(1)}2^{o(n)}$ [BLT15; JLL16]. Subset-Sum can also be solved in pseudopolynomial time, for instance in $O(nm)$ by a textbook dynamic programming approach, which was improved to a highly elegant $\widetilde{O}(n+m)$ randomized algorithm by Bringmann [Bri17]. However, assuming the Strong Exponential Time Hypothesis (SETH), it can be shown that for all $\epsilon > 0$, there exists $\delta > 0$, such that Subset-Sum cannot be computed in time $O(m^{1-\epsilon}2^{n\delta})$ [ABHS19]. On a quantum computer, meet-in-the-middle can be combined with quantum search to solve Subset-Sum in time $\widetilde{O}(2^{n/3})$. A modular version of Subset-Sum can be similarly defined:

Problem 2 (Modular Subset-Sum)**.** Given a multiset $\{a_1, \ldots, a_n\}$ of positive integers, a target integer m and a modulus q, find a subset $S \subseteq [n]$ such that $\sum_{i \in S} a_i \equiv m \pmod{q}$.

The $\widetilde{O}(2^{n/2})$ classical and $\widetilde{O}(2^{n/3})$ quantum meet-in-the-middle algorithms, as well as the classical $O(nm)$ dynamic programming algorithm can be used to solve Modular Subset-Sum with the same running times, by replacing regular addition with modular addition. While the $\widetilde{O}(n+m)$ algorithm of Bringmann does not immediately give rise to an $\widetilde{O}(q)$ algorithm for Modular Subset-Sum, several recent algorithms have achieved this complexity [ABJ+19; ABB+21; CI21]. Also, SETH implies that for all $\epsilon > 0$, there exists $\delta > 0$, such that Modular Subset-Sum cannot be computed in time $O(q^{1-\epsilon}2^{n\delta})$ because an instance of Subset-Sum where each $a_i < m$ is a special case of Modular Subset-Sum when we choose $q = nm$.

1.1 Some variants of Subset-Sum

Subset-Sum has several close relatives we will be concerned with in this paper. First among these is Equal-Sums, introduced by Woeginger and Yu [WY92], where one must decide if a set of n positive integers contains two disjoint subsets whose elements sum to the same value:

Problem 3 (Equal-Sums [WY92])**.** Given a set $\{a_1, \ldots, a_n\}$ of positive integers, find two distinct subsets $S_1, S_2 \subseteq [n]$ such that $\sum_{i \in S_1} a_i = \sum_{i \in S_2} a_i$. In inner product notation, we are looking for a nonzero vector $\bar{e} \in \{-1,0,1\}^n$ such that $\bar{a} \cdot \bar{e} = 0$.

The folklore classical algorithm [Woe08] for Equal-Sums runs in time $\widetilde{O}(3^{n/2}) \leq O(2^{0.793n})$, and is also based on a meet-in-the-middle approach. In the classical case, we arbitrarily partition the input into two sets of the same size, giving rise to vectors $\bar{a}_1, \bar{a}_2 \in \mathbb{N}^{n/2}$. Then we compute and sort the possible $3^{n/2}$ values $\bar{a}_1 \cdot \bar{e}$, for $\bar{e} \in \{-1,0,1\}^{n/2}$. Finally, we compute the possible $3^{n/2}$ values of the form $\bar{a}_2 \cdot \bar{e}$ and, for each value, check via binary search if it has a collision (i.e. an item of the same value) in the first set of values. In the quantum case, we use a different balancing, dividing the input into a set of size $n/3$ and a set of size $2n/3$, and then use quantum search over the larger set to find a collision. This folklore quantum algorithm has a running time of $\widetilde{O}(3^{n/3}) \leq O(2^{0.529n})$. The classical running time of Equal-Sums was reduced in a recent work by Mucha et al. [MNPW19] to $O(2^{0.773n})$, and it is an open problem whether this can be further improved. The modular version of Equal-Sums is defined as:

Problem 4 (Modular Equal-Sums)**.** Given a set $\{a_1, \ldots, a_n\}$ of positive integers and a modulus q, find two distinct subsets $S_1, S_2 \subseteq [n]$ such that $\sum_{i \in S_1} a_i \equiv \sum_{i \in S_2} a_i \pmod q$.

Similarly to Subset-Sum, the $O(2^{0.793n})$ time meet-in-the-middle algorithm for Equal-Sums gives rise to an algorithm of the same time for Modular Equal-Sums. Moreover, we can suppose that $q \geq 2^n$, because otherwise we can just consider $a_1, \ldots, a_k$ from the input, where k satisfies $2^{k-1} \leq q < 2^k$. By the pigeonhole principle, such an instance has a solution which we will show (see Theorem 6.4) can be found in time $\widetilde{O}(2^{k/2})$. Thus, Modular Equal-Sums can always be solved in time $O(q^{0.793})$ classically. Intriguingly, when expressed as a function of n, faster algorithms are known for both Subset-Sum and Modular Subset-Sum than for Equal-Sums and Modular Equal-Sums, respectively, whereas expressed as a function of q (or as a function of $\sum_{i=1}^n a_i$ in the non-modular cases), the situation is the opposite. This holds both classically and quantumly.

A natural generalization of Subset-Sum is to allow each item in the input set to be used more than once in the summation, where the maximum number of times each item can be used is specified as part of the input to the problem. This is the analog of bounded knapsack, a well-studied problem in the literature (see for example [KPP04]). In particular, we will study the case when every item can be used at most twice.

Problem 5 (2-Subset-Sum)**.** Given a multiset $\{a_1, \ldots, a_n\}$ of positive integers and a target integer $0 < m < 2\sum_{i=1}^n a_i$, find a vector $\bar{e} \in \{0,1,2\}^n$, such that $\bar{a} \cdot \bar{e} = m$.

There is a natural variant of Subset-Sum that generalizes both Equal-Sums and 2-Subset-Sum. We call this variant Shifted-Sums, whose investigation is the main subject of this paper.

Problem 6 (Shifted-Sums)**.** Given a multiset $\{a_1, \ldots, a_n\}$ of positive integers and an integer $0 \leq s < \sum_{i=1}^n a_i$, find two distinct subsets $S_1, S_2 \subseteq [n]$ such that $\sum_{i \in S_1} a_i = s + \sum_{i \in S_2} a_i$.

The condition $S_1 \neq S_2$ is necessary in the case $s = 0$ to exclude the trivial solutions $S_1 = S_2$. The problem Equal-Sums is a special case of Shifted-Sums in this case, and it is easy to show (see Proposition 2.3) that 2-Subset-Sum can also be reduced to Shifted-Sums without increasing the size of the input. This means that any algorithm for Shifted-Sums automatically gives rise to an algorithm of the same complexity for Equal-Sums and 2-Subset-Sum, and therefore we focus on constructing classical and quantum algorithms for Shifted-Sums. We also consider the modular version of Shifted-Sums:

Problem 7 (Modular Shifted-Sums)**.** Given a multiset $\{a_1, \ldots, a_n\}$ of positive integers, an integer $0 \leq s < \sum_{i=1}^n a_i$ and a modulus q, find two distinct subsets $S_1, S_2 \subseteq [n]$ such that $\sum_{i \in S_1} a_i \equiv s + \sum_{i \in S_2} a_i \pmod q$.

We additionally study the following variant of Equal-Sums where, by the pigeonhole principle, a solution is guaranteed to exist. This search problem is total in the sense that its decision version is trivial because the answer is always 'yes'. Such problems belong to the complexity class TFNP [MP91] consisting of NP-search problems with total relations. Problems in TFNP cannot be NP-hard unless NP equals co-NP. More precisely, the following two problems belong to the Polynomial Pigeonhole Principle complexity class PPP, defined by Papadimitriou [Pap90], where the totality of the problem is syntactically guaranteed by the pigeonhole principle.

Problem 8 (Pigeonhole Equal-Sums)**.** Given a set $\{a_1, \ldots, a_n\}$ of positive integers such that $\sum_{i=1}^n a_i < 2^n - 1$, find two distinct subsets $S_1, S_2 \subseteq [n]$ such that $\sum_{i \in S_1} a_i = \sum_{i \in S_2} a_i$.

There are 2^n subsets $S \subseteq [n]$. Since they all verify $0 \leq \sum_{i \in S} a_i \leq 2^n - 2$ there must exist two distinct subsets S_1, S_2 that sum to the same value, according to the pigeonhole principle. The modular version of Pigeonhole Equal-Sums similarly belongs to the class PPP:

Problem 9 (Pigeonhole Modular Equal-Sums)**.** Given a set $\{a_1, \ldots, a_n\}$ of positive integers and a modulus q such that $q \leq 2^n - 1$, find two distinct subsets $S_1, S_2 \subseteq [n]$ such that $\sum_{i \in S_1} a_i \equiv \sum_{i \in S_2} a_i \pmod{q}$.

Observe that Pigeonhole Equal-Sums is a special case of Pigeonhole Modular Equal-Sums when $q = 2^n - 1$.

1.2 Our contributions and techniques

We give new classical and quantum algorithms for Subset-Sum and several closely related problems defined in the previous section. Our results are succinctly stated below and summarized in Table 1. The algorithms for Subset-Sum achieve the same complexity as the currently best-known algorithms based on the meet-in-the-middle method[1]. Our quantum algorithm for Shifted-Sums (and for its special cases of Equal-Sums and 2-Subset-Sum) improves on the currently best-known $O(2^{0.529n})$ quantum algorithm for these problems, which is also based on meet-in-the-middle. Our quantum algorithm for Pigeonhole Equal-Sums further improves, in this special case, on our algorithm for general Equal-Sums. We also initiate the study of the Pigeonhole Equal-Sums problem (and its modular variant) in the classical setting, where we obtain a better complexity than what was known before for the general Equal-Sums problem.

Theorems 4.1, 4.2 (Restated)**.** *There are representation-technique-based classical and quantum algorithms for* Subset-Sum *that run in time $\widetilde{O}(2^{n/2})$ and $\widetilde{O}(2^{n/3})$, respectively.*

Theorems 5.2, A.3 (Restated)**.** *There are classical and quantum algorithms for* Shifted-Sums *that run in time $O(2^{0.773n})$ and $O(2^{0.504n})$, respectively.*

Theorem 6.3 (Restated)**.** *There is a quantum algorithm for* Pigeonhole Equal-Sums *that runs in time $\widetilde{O}(2^{2n/5})$.*

Theorems 6.2, 6.4 (Restated)**.** *There are classical deterministic algorithms for* Pigeonhole Equal-Sums *and* Pigeonhole Modular Equal-Sums *that run in time $\widetilde{O}(2^{n/2})$.*

At a high level, all of our algorithms use a *representation technique* approach. While this technique was originally designed to solve Subset-Sum when the instances are drawn from some specific distribution [HJ10], here we follow the path of Mucha et al. [MNPW19] and use it in a worst-case analysis. Among our three main algorithms, the quantization of this technique for Shifted-Sums is the most challenging. We will therefore explain first, via this algorithm, the difficulties we had to address and the methods we used to tackle them.

Shifted-Sums. The representation technique approach for Shifted-Sums consists first of selecting a random prime $p \in \{2^{bn}, \ldots, 2^{bn+1}\}$, where $b \in (0, 1)$ is some appropriate constant, and a random integer $k \in \{0, \ldots, p-1\}$. Then we consider the random *bin* $T_{p,k}$, defined as

$$T_{p,k} = \Big\{S \subseteq \{1, \ldots, n\} : \sum_{i \in S} a_i \equiv k \pmod{p}\Big\},$$

and we search that bin and $T_{p,(k-s) \bmod p}$ for a colliding solution (i.e. a pair of sets $(S_1, S_2) \in T_{p,k} \times T_{p,(k-s) \bmod p}$ such that $\sum_{i \in S_1} a_i = s + \sum_{i \in S_2} a_i$). The choice of the bin size (which,

[1]After completion of this work, it was pointed out to us by an anonymous referee that a classical algorithm for Subset-Sum, similar to ours, was sketched in [AKKN16].

	Classical	Quantum
SUBSET-SUM	$2^{n/2}$ [HS74; AKKN16], [Thm. 4.2]	$2^{n/3}$ [BJLM13], [Thm. 4.1]
SHIFTED-SUMS	$2^{0.773n}$ [MNPW19], [Thm. A.3]	$2^{0.504n}$ [Thms. 5.2, B.2]
PIGEONHOLE EQUAL-SUMS	$2^{n/2}$ [Thm. 6.2]	$2^{2n/5}$ [Thm. 6.3]
PIGEONHOLE MODULAR EQUAL-SUMS	$2^{n/2}$ [Thm. 6.4]	–

Table 1: Best-known classical and quantum running times for variants of SUBSET-SUM. Our results are indicated by reference to the corresponding theorems in this paper. The $\widetilde{O}(\cdot)$ notation is implied for all running times. For SUBSET-SUM our dynamic programming based results have the same time complexity as the best previous algorithms that were based on meet-in-the-middle. The results of [MNPW19] are for EQUAL-SUMS, a special case of SHIFTED-SUMS. In Appendix A, we give an extension of their algorithm to SHIFTED-SUMS.

on average, is roughly $2^{(1-b)n}$) should balance two opposing requirements: the bins should be sufficiently large to contain a solution and also sufficiently small to keep the cost of collision search low.

To satisfy the above two requirements, our algorithm uses the concept of a *maximum solution*. This is the maximum of $|S_1| + |S_2|$, when S_1, S_2 are disjoint and form a solution. Let this maximum solution size be ℓn, for some $\ell \in (0, 1)$. The algorithm consists of two different procedures, designed to handle different maximum solution sizes. For ℓ close to 0 or close to 1, the quantization of the meet-in-the-middle method adapted to solutions of size ℓn is used because it performs better. In this case, the quantization does not present any particular difficulties: it is a straightforward application of quantum search with the appropriate balancing. We therefore focus the discussion on the representation technique procedure used for values of ℓ away from 0 or 1. When S_1, S_2 form a maximum solution of size ℓn then, for every set $X \subseteq \overline{S_1 \cup S_2}$ in the complement of the solution, the pairs $S_1 \cup X, S_2 \cup X$ also form a solution, and all these solutions have different values (see Lemma 5.4). This makes it possible to bound from below, not only the number of solutions, but also the number of *solution values* by $2^{(1-\ell)n}$, which makes the use of the representation technique successful.

The most immediate way to quantize the procedure is to replace classical collision finding with the quantum element distinctness algorithm of Ambainis [Amb07]. However, in a straightforward application of this algorithm we face a difficulty. For concreteness, we explain this when $\ell = 3/5$. In that case, by the above, the total number of solutions with different values is at least $2^{2n/5}$. This is handy for applying quantum element distinctness: we can select a random prime $p \in \{2^{2n/5}, \ldots, 2^{2n/5+1}\}$ and expect to have a solution in the random bin $T_{p,k}$ with reasonable probability. The expected size $|T_{p,k}|$ of the bin is about $2^{3n/5}$, and therefore the running time of Ambainis' algorithm should be of the order of $|T_{p,k}|^{2/3}$ which is also about $2^{2n/5}$. However, the quantum element distinctness algorithm requires us to perform queries to $T_{p,k}$. That is, for some indexing $T_{p,k} = \{S_1, \ldots, S_{|T_{p,k}|}\}$ of the elements of $T_{p,k}$, we need to implement the oracle $O_{T_{p,k}}|I\rangle|0\rangle = |I\rangle|S_I\rangle$, where $1 \leq I \leq |T_{p,k}|$. In other words, given $1 \leq I \leq |T_{p,k}|$, we have to be able to find the Ith element in $T_{p,k}$ (for some ordering of that set). In the usual description of the element distinctness algorithm there is a simple way to do that (for example, the set over which the algorithm is run is just a set of consecutive integers). However, finding a simple bijection among the first $|T_{p,k}|$ integers and $T_{p,k}$ is not a trivial task. Unlike in the classical case, explicitly enumerating $T_{p,k}$ is not an option because this would take too long, requiring about $2^{3n/5}$ time steps. Instead, we use *dynamic programming* to compute the table of cardinalities,

$$t_p[i,j] = \left|\left\{S \subseteq \{1,\dots,i\} : \sum_{s\in S} a_s \equiv j \pmod p\right\}\right|.$$

Computing the cardinality of the bins is cheaper than computing their contents, and can be done in time $O(n^2 p) = \widetilde{O}(2^{2n/5})$. Crucially, once the table is constructed, one can deduce the paths through it that led to $t_p[n,k] = |T_{p,k}|$, in order to find each element of $T_{p,k}$ in time $O(n^2)$. More precisely, we define a particular strict total order $\prec$ over $\mathcal{P}([n])$ and prove:

Theorem 3.5 (Restated)**.** *Let $T_{p,k}$ be enumerated as $T_{p,k} = \{S_1,\dots,S_{|T_{p,k}|}\}$ where $S_1 \prec \cdots \prec S_{|T_{p,k}|}$. Given any integer $I \in \{1,\dots,|T_{p,k}|\}$ and random access to the elements of the table t_p, the set S_I can be computed in time $O(n^2)$.*

This novel data structure will be used in our algorithms for Subset-Sum, Shifted-Sums and Pigeonhole Equal-Sums. We now describe the additional quantum tools we use for Shifted-Sums. The algorithm randomly chooses a bin of size about $2^{(1-b)n}$ where b is defined differently depending on whether ℓ is above or below $3/5$, as different quantum tools are required in these two regions. When $\ell \le 3/5$, with high probability a random bin will contain multiple solutions from which we can profit. To that end, we construct a quantum algorithm for finding a pair marked by a binary relation $\mathcal{R}(x,y) := \mathbb{1}_{f(x)=g(y)}$ that tests if two values $f(x)$ and $g(y)$ are equal or not. Our algorithm generalizes the quantum element distinctness [Amb07] and claw finding [Tan09] algorithms to the case of multiple marked pairs. Using an appropriate variant of the birthday paradox (see Lemma 2.7) we prove:

Theorem 2.8 (Quantum pair finding - Restated)**.** *Consider two sets of $N \le M$ elements, respectively, and an evaluation function on each set. Suppose that there are K disjoint pairs in the product of the two sets such that in each pair the elements evaluate to the same value. There is a quantum algorithm that finds such a pair in time $\widetilde{O}\big((NM/K)^{1/3}\big)$ if $N \le M \le KN^2$ and $\widetilde{O}\big((M/K)^{1/2}\big)$ if $M \ge KN^2$.*

The best complexity when $\ell \le 3/5$ is then obtained by choosing the bin size parameter b as a function of ℓ, which balances the cost of the construction of the dynamic programming table and the quantum pair finding. When $\ell > 3/5$, choosing a bin size $2^{(1-b)n}$, for $b \le 1-\ell$, guarantees that a random bin contains at least one solution with high probability. However, a better running time at first seems to be achievable by the following argument: Choose $b > 1-\ell$, for which there is an exponentially small probability that a random bin contains a solution, and use amplitude amplification to boost the success probability. Balancing again the dynamic programming and quantum pair finding costs would then give an optimal bin size of $2^{3n/5}$, independent of ℓ. However, this argument contains a subtlety. Standard amplitude amplification requires that the random bin $T_{p,k}$ simultaneously satisfies two conditions: besides containing a solution, it should also have size close to the expected size of about $2^{(1-b)n}$. But there is no guarantee that these two events coincide, and a priori it could be that the exponentially small fraction of $T_{p,k}$ containing a solution also happens to have sizes that far exceed the expectation. Fortunately, by carefully bounding the expectation of the product of bin sizes, we can use the variable-time amplitude amplification algorithm of Ambainis [Amb12], and achieve the same running time as given by the above argument. We believe that this is a nice and natural application of this method. The running time of our algorithm for Shifted-Sums, as a function of ℓ, is shown in Fig. 1.

Pigeonhole Equal-Sums. This problem can be solved by any (classical or quantum) algorithm that solves the general Equal-Sums (or Shifted-Sums) problem. However, one can make use of the explicit promise of $a_1 + \cdots + a_n < 2^n - 1$ to design faster algorithms than provided for by the general case when $\ell > 3/5$. Indeed, by the pigeonhole principle, for *any* value

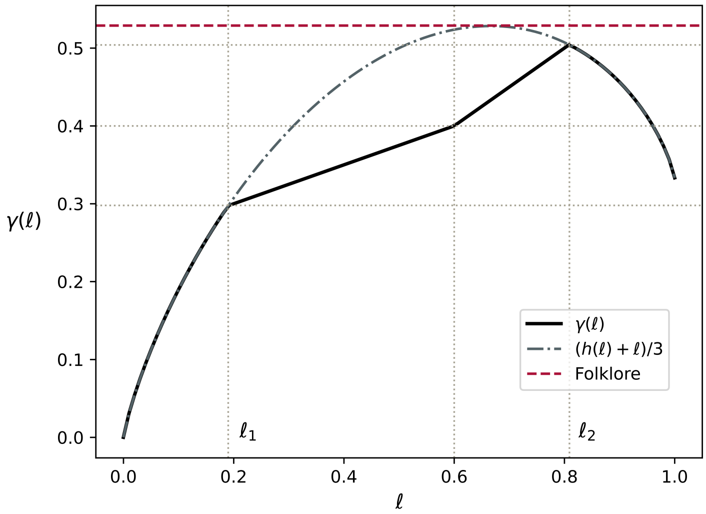

Figure 1: Running time exponent $\gamma(\ell)$ of the quantum SHIFTED-SUMS algorithm, as a function of the maximum solution ratio ℓ (see Theorem 5.2). The maximum value of $\gamma(\ell)$ is ≈ 0.504 and it occurs at $\ell = \ell_2 \approx 0.809$. For reference, the curve $(h(\ell) + \ell)/3$ corresponding to Theorem 5.7 is plotted for all values of ℓ, as is the value of 0.529 corresponding to the exponent of the folklore (quantized) meet-in-the-middle algorithm, as applied to SHIFTED-SUMS.

of p, if a bin $T_{p,k}$ has a size larger than $2^n/p$ then it must contain a solution (see Lemma 6.1). Moreover, there must exist at least one such oversized bin. The array t_p can now be constructed both for *locating the index* k of one oversized bin and searching for a solution in it. We thus obtain a classical algorithm running in time $\widetilde{O}(p + 2^n/p)$, and a quantum algorithm running in time $\widetilde{O}(p + (2^n/p)^{2/3})$. These two quantities are minimized by *deterministically* choosing $p = 2^{n/2}$ and $p = 2^{2n/5}$ respectively (see Section 6.1).

PIGEONHOLE MODULAR EQUAL-SUMS. In general, we do not know how to extend our techniques to the problems modulo some integer q. A natural approach would be to consider the bins $T_{p,k} = \left\{S \subseteq \{1, \ldots, n\} : (\sum_{i \in S} a_i \bmod q) \equiv k \pmod p\right\}$, but it is unclear how to compute the corresponding table t_p efficiently. We give a solution to this problem for PIGEONHOLE MODULAR EQUAL-SUMS that works in the classical setting only (see Section 6.2).

1.3 Related works

The closest work to our contribution is the paper of Mucha et al. [MNPW19] solving EQUAL-SUMS classically in time $O(2^{0.773n})$. Their algorithm and ours use the same two basic procedures, based respectively on the meet-in-the-middle method and the representation technique. Let us point out some of the differences. Unlike our algorithm that is based on the concept of the size of a maximum solution, the classical algorithm is analysed as a function of a *minimum size* solution, defined as $|S_1| + |S_2|$, where this sum is minimized over all solutions. The use the classical algorithm makes from a minimum solution S_1, S_2 of size ℓn is that when $\ell > 1/2$ the number of *solution values* can be bounded from below by $2^{(1-\ell)n}$. However, this does not hold for 2-SUBSET-SUM when ℓn is the size of a minimum solution, but *is* valid for both EQUAL-SUMS and 2-SUBSET-SUM when it is the size of a maximum solution. Another difference with [MNPW19] is that their classical representation technique algorithm always samples p from the same set $\{2^{(1-\ell)n}, \ldots, 2^{(1-\ell)n+1}\}$, while we randomly choose $p \in \{2^{bn}, \ldots, 2^{bn+1}\}$ where b is defined differently depending on in which of two distinct regions ℓ lies. This makes it possible to use different quantum techniques in these two regions.

The representation technique was designed by Howgrave-Graham and Joux [HJ10] to solve random Subset-Sum instances under some hypotheses (heuristics) about how such instances behave during the run of the algorithm. The idea is to decompose a single solution to the initial problem into many distinct decompositions of a sum of half-solutions. To compensate for this blow-up, an additional linear constraint is added to select approximately one of these decompositions. Under some rather strong assumptions, which are satisfied for a large fraction of randomly chosen instances, [HJ10] can solve Subset-Sum instances in time $O(2^{0.337n})$. Since then, several variants of this classical method have been proposed [BCJ11; EM20; BBSS20; CJRS22], while others have investigated quantum algorithms based on the representation technique. Bernstein et al. [BJLM13] improved on [HJ10] using quantum walks, and their algorithm (again under some hypotheses) runs in time $O(2^{0.242n})$. Further quantum improvements were made in this context by [HM18] and [BBSS20]. However, we emphasize that the algorithms in all these papers work for random inputs generated from some distributions. The paper [MNPW19] gave the first classical algorithm based on the representation technique that works for worst-case inputs and with proven bounds. To our knowledge, for worst-case inputs with provable guarantees, the first quantum algorithm based on the representation technique is given in our work.

Dynamic programming is notoriously hard to quantize, with a key obstacle being the intrinsically sequential way in which the solution to a large problem is constructed from the solutions to smaller subproblems. Certain basic dynamic programming algorithms can be trivially accelerated by quantum search or minimum finding (see, e.g. [Abb19]) but beyond that few other quantum improvements are known. One notable exception is the work of Ambainis et al. [ABI+19] who gave faster quantum algorithms for several NP-hard problems for which the best classical algorithms use dynamic programming. Their algorithms precompute solutions for smaller instances via dynamic programming and then use non-trivial quantum search recursively on the rest of the problem. In our work, the dynamic programming subroutine that we use is classical (although for Shifted-Sums it is performed in superposition) and the sequential nature of the process is therefore not an issue. Rather than using quantum computing to accelerate classical dynamic programming, we instead use dynamic programming to enable fast queries, required for quantum search and pair finding, to be performed on complicated sets.

The class PPP is arguably less studied than other syntactically definable subclasses of TFNP, such as PLS (Polynomial Local Search) and PPA (Polynomial Parity Argument), and it is not known whether Pigeonhole Equal-Sums is complete in PPP. In fact, the first complete problem for the class was only identified relatively recently [SZZ18]. Our results for Pigeonhole Equal-Sums suggest that the problem is indeed simpler to solve than Equal-Sums. In spirit, a similar result was obtained in [BST02] where it was shown that, for an optimization problem closely related to Equal-Sums, better approximation schemes can be obtained for instances with guaranteed solutions.

1.4 Open problems

We suggest two directions for future work on the modular versions of the problems studied in this paper:

1. Quantization of Pigeonhole Modular Equal-Sums. That is, can we construct a quantum algorithm improving on the classical running time given in Theorem 6.4?
2. Can we prove a modular version of the algorithm of [MNPW19]? That is, can Modular Equal-Sums be solved classically in time $O(q^{0.773})$?

1.5 Structure of the paper

The paper is organised as follows. In Section 2 we define the quantum computational model and some basic notations, and we state several facts, propositions and algorithmic tools (such as our quantum pair finding algorithm) used in subsequent sections. In Section 3 we introduce the dynamic programming data structure and show how it can be used to implement fast subset-sum queries. We present a simple application of this data structure to the SUBSET-SUM problem in Section 4, where we obtain new classical and quantum algorithms achieving the same complexity as the best-known algorithms based on meet-in-the-middle. The main applications are described in Section 5 for the SHIFTED-SUMS problem, and in Section 6 for the pigeonhole variants of EQUAL-SUMS. Finally, we adapt our results to the classical SHIFTED-SUMS problem in Appendix A, and we describe an alternative quantum algorithm for EQUAL-SUMS whose complexity is parametrized by the minimum solution ratio in Appendix B.

2 Preliminaries

2.1 Notations

We use the $\widetilde{O}(x)$ and $\widetilde{\Omega}(x)$ notations to hide factors that are polylogarithmic in the argument x. For integers $0 \leq m < n$, we denote by $[m\,..\,n]$ the set $\{m, m+1, \dots n\}$, and by $[n]$ the set $[1\,..\,n]$. For sets $S, S' \subseteq [n]$ we denote by $\bar{S}$ the complement $[n] \setminus S$, and by $S \Delta S'$ the symmetric difference of S and S'. Given a multiset $A = \{a_1, \dots, a_n\}$ and a subset $S \subseteq [n]$ we denote $\Sigma_A(S) := \sum_{i \in S} a_i$. When the set A is clear from the context, we will omit the subscript and simply denote the subset sum by $\Sigma(S)$. The power set of $[n]$ will be denoted by $\mathcal{P}([n]) := \{S : S \subseteq [n]\}$. For arbitrary integers a and b and a modulus p we say that a is congruent to b modulo p, and we write $a \equiv b \pmod p$ or $a \equiv_p b$ if $a - b$ is divisible by p. By $a \bmod p$ we denote the unique integer in $\{0, \dots, p-1\}$ that is congruent to a modulo p. The binary entropy function will be denoted by $h(x) = -x \log_2(x) - (1-x)\log_2(1-x)$.

2.2 Quantum computational model

Similar to previous works on quantum element distinctness [Amb07], quantum dynamic programming [ABI+19] and quantum subset sum algorithms [BJLM13], in our quantum algorithm running time analysis we assume the standard circuit model (where computational time corresponds to the number of single and two-qubit gates) augmented with random access to quantum memory. That is, coherent access to any element of an m-qubit array can be performed in time polylogarithmic in m. Note that fully quantum memory is required in Algorithm 4 for SHIFTED-SUMS since multiple bins $T_{p,k}$ must be computed and stored in superposition. On the other hand, Algorithm 2 for SUBSET-SUM only requires a single bin $T_{p,k}$ to be searched. Thus, while the memory cells must be accessed in superposition, the data that each cell holds is classical.

2.3 Basic Facts

Fact 2.1 ([Gal68], page 530)**.** *For every constant $\ell \in (0,1)$ and for every large enough integer n, the following bounds hold:* $\frac{2^{nh(\ell)}}{\sqrt{8n\ell(1-\ell)}} \leq \binom{n}{\ell n} < \frac{2^{nh(\ell)}}{\sqrt{2\pi n\ell(1-\ell)}}$.

Fact 2.2. *Let $b > 0$ be a constant, n a large enough integer and $a_1 \neq a_2$ two integers. Then, for a random prime $p \in [2^{bn}\,..\,2^{bn+1}]$ we have $\Pr_p[a_1 \equiv_p a_2] \leq \frac{\log(|a_1 - a_2|)}{2^{bn}}$.*

Proof. The number of primes that belong to the interval $[2^{bn}\,..\,2^{bn+1}]$ is at least $2^{bn}/bn$ for n large enough (see [HW75, p.371]). Moreover, there are at most $\frac{\log(|a_1-a_2|)}{bn}$ prime numbers larger than 2^{bn} that divide $a_1 - a_2$. The result follows by a union bound. $\square$

2.4 Size preserving reductions

We say that a polynomial time reduction between two problems is *size preserving* if it preserves the number of input items. The following propositions justify the assumption in our algorithms that the input items are bounded by 2^{4n}.

Proposition 2.3. *There exist size preserving reductions from* Equal-Sums *to* Shifted-Sums *and from* 2-Subset-Sum *to* Shifted-Sums.

Proof. In fact, the claimed reductions are not only size preserving, but also keep the input items $\{a_1, \ldots, a_n\}$ intact. Equal-Sums is simply the restriction of Shifted-Sums to the case $s = 0$. Consider an instance of 2-Subset-Sum, and set $W = \sum_{i=1}^{n} a_i$. We can suppose, without loss of generality, that $W < m < 2W$ because $\sum_{i=1}^{n} a_i \cdot e_i = m$ if and only if $\sum_{i=1}^{n} a_i \cdot (2 - e_i) = 2W - m$, and the $m = W$ case is trivial.

We claim that $\bar{a} = (a_1, \ldots, a_n), m$ is a positive instance of 2-Subset-Sum if and only if $\bar{a}, m - W$ is a positive instance of Shifted-Sums. For a vector $\bar{e} \in \{0, 1, 2\}^n$, define $e_i^1 = 1$ if $e_i = 2$ and 0 otherwise, and similarly define $e_i^2 = 1$ if $e_i = 0$ and 0 otherwise. Then $\bar{e}^1, \bar{e}^1 \in \{0, 1\}^n$ and $e_i = e_i^1 - e_i^2 + 1$ implying $\bar{a} \cdot \bar{e} = m$ if and only if $\bar{a} \cdot \bar{e}^1 = \bar{a} \cdot \bar{e}^2 + m - W$. Therefore $\bar{e}$ is a solution of 2-Subset-Sum if and only if $\bar{a}, m - W$ is a solution of Shifted-Sums. ☐

Given a positive instance of 2-Subset-Sum, the above reduction produces a positive instance of Shifted-Sums with $s \neq 0$. It is easy to check that the reduction also works in the reverse direction. Taken together, this implies that Shifted-Sums with $s \neq 0$ is just a reformulation of 2-Subset-Sum and, as we have already observed, with $s = 0$ is exactly Equal-Sums.

Proposition 2.4. *There exists a probabilistic size preserving reduction from* Shifted-Sums *to an instance of* Shifted-Sums *where the input items satisfy* $\sum_{i=1}^{n} a_i < 2^{4n}$. *A similar statement holds also for* Subset-Sum.

Proof. We prove the statement for Shifted-Sums, the proof for Subset-Sum is analogous. Let $a_1, \ldots, a_n, s$ be an instance of Shifted-Sums. Without loss of generality, we can suppose that $\sum_{i=1}^{n} a_i < 2^{2^n}$ because otherwise we can solve the instance in polynomial time of the input size. We choose a random prime $p \in \{2^{4n-1}, \ldots, 2^{4n}\}$ and define the reduced instance by $s' = s \bmod p$ and $a_i' = a_i \bmod p$. It is obvious that if the original instance has a solution then so does the reduced instance. We now claim that if the original instance does not have a solution, then the reduced instance has a solution only with probability at most $O(2^{-n})$. This is because a random prime p divides $\sum_{i \in S_1} a_i - \sum_{i \in S_2} a_i - s$ with probability $O(\log(2^{2^n})/2^{4n})$ when (S_1, S_2) is not a solution (Fact 2.2), and there are $O(2^{2n})$ such pairs to consider. ☐

2.5 Quantum algorithms

We use the following generalization of Grover's search to the case of an unknown number of solutions.

Fact 2.5 (Quantum search, Theorem 3 in [BBHT98]). *Consider a function* $f : [N] \to \{0, 1\}$ *with an unknown number* $K = |f^{-1}(1)|$ *of marked items. Suppose that* f *can be evaluated in time* τ. *Then, the quantum search algorithm finds a marked item in* f *in expected time* $\widetilde{O}(\sqrt{N/K} \cdot \tau)$.

Given a classical subroutine with stopping time τ that returns a marked item with probability ρ, we can convert it into a constant success probability algorithm with expected running time $O(\mathbb{E}[\tau]/\rho)$ by repeating it $O(1/\rho)$ times. Ambainis proved a similar result for the case of quantum subroutines, with a dependence on the second moment of the stopping time τ, and a Grover-like speed-up for the dependence on ρ.

Fact 2.6 (VARIABLE-TIME AMPLITUDE AMPLIFICATION, Theorem 2 in [Amb12]). *Let $\mathcal{A}$ be a quantum algorithm that looks for a marked element in some set. Let τ be the random variable corresponding to the stopping time of the algorithm, and let ρ be its success probability. Then the variable-time amplitude amplification algorithm finds a marked element in the above set with constant success probability in maximum time $\widetilde{O}(\sqrt{\mathbb{E}[\tau^2]/\rho})$.*

The next result is a variant of the Birthday paradox over a product space $[N] \times [M]$, where at least K *disjoint* pairs are marked by some binary relation $\mathcal{R}$. Two pairs (x, y) and (x', y') are said to be disjoint if $x \neq x'$ and $y \neq y'$. The disjointness assumption is made to simplify the analysis and will be satisfied in our applications.

Lemma 2.7 (VARIANT OF THE BIRTHDAY PARADOX). *Consider three integers $1 \leq K \leq N \leq M$. Let $\mathcal{R} : [N] \times [M] \to \{0, 1\}$ be a binary relation such that there exist at least K mutually disjoint pairs $(x_1, y_1), \ldots, (x_K, y_K) \in [N] \times [M]$ with $\mathcal{R}(x_k, y_k) = 1$ for all $k \in [K]$. Given an integer $r \leq O(\sqrt{NM/K})$, define $\epsilon(r)$ to be the probability of obtaining both elements from at least one marked pair when r numbers from $[N]$ and r numbers from $[M]$ are chosen independently and uniformly at random. Then, $\epsilon(r) \geq \Omega\left(\frac{r^2 K}{NM}\right)$.*

Proof. Fix any K disjoint marked pairs $(x_1, y_1), \ldots, (x_K, y_K)$. Let $X_1, \ldots, X_r$ (resp. $Y_1, \ldots, Y_r$) be r independent and uniformly distributed random variables over $[N]$ (resp. $[M]$). For any indices i, j, let $Z_{i,j}$ denote the binary random variable that takes value 1 if $\{X_i, X_j\}$ is one of the K fixed pairs, and set $Z = \sum_{i,j} Z_{i,j}$. By definition, we have $\epsilon(r) \geq \Pr[Z \neq 0]$. We lower bound this quantity by using the inclusion-exclusion principle,

$$\epsilon(r) \geq \sum_{i,j} \Pr(Z_{i,j} = 1) - \frac{1}{2} \sum_{(i,j) \neq (k,\ell)} \Pr(Z_{i,j} = 1 \wedge Z_{k,\ell} = 1).$$

The first term on the right-hand side is equal to $\sum_{i,j} \Pr(Z_{i,j} = 1) = \frac{r^2 K}{NM}$. For the second term, the analysis depends on whether the indices i, k and j, ℓ are distinct or not. If they are distinct then $\Pr(Z_{i,j} = 1 \wedge Z_{k,\ell} = 1) = \frac{K^2}{(NM)^2}$ since $Z_{i,j}$ and $Z_{k,\ell}$ are independent. Otherwise, suppose for instance that $i = k$. Since the K fixed pairs are disjoint, we have $\Pr(Z_{i,j} = 1 \wedge Z_{i,\ell} = 1) = \Pr(Z_{i,j} = 1 \wedge Y_j = Y_\ell) = \frac{K}{NM^2}$. Finally, there are $4\binom{r}{2}^2$ ways of choosing the indices i, j, k, ℓ when $i \neq k$ and $j \neq \ell$, and $4r\binom{r}{2}$ ways when $i = k$ or $j = \ell$. By putting everything together we obtain that, $\epsilon(r) \geq \frac{r^2 K}{NM} - 2\binom{r}{2}^2 \frac{K^2}{(NM)^2} - 2r\binom{r}{2} \frac{K}{NM^2} \geq \Omega\left(\frac{r^2 K}{NM}\right)$. ☐

We use the above result to construct a quantum algorithm for finding a marked pair when the relation $\mathcal{R}(x, y)$ is determined by checking if two underlying values $f(x)$ and $g(y)$ are equal or not. Our analysis essentially generalizes the quantum element distinctness [Amb07] and claw finding [Tan09] algorithms to the case of $K > 1$.

Theorem 2.8 (QUANTUM PAIR FINDING). *There is a bounded-error quantum algorithm with the following properties. Consider four integers $1 \leq K \leq N \leq M \leq R$ with $R \leq N^{O(1)}$. Let $f : [N] \to [R]$ and $g : [M] \to [R]$ be two functions that can be evaluated in time τ. Define $\mathcal{R} : [N] \times [M] \to \{0, 1\}$ to be any of the two following binary relations:*

1. *$\mathcal{R}(x, y) = 1$ if and only if $f(x) = g(y)$.*

2. *$\mathcal{R}(x, y) = 1$ if and only if $f(x) = g(y)$ and $x \neq y$.*

Suppose that there exist at least K mutually disjoint pairs $(x, y) \in [N] \times [M]$ such that $\mathcal{R}(x, y) = 1$. Then, the algorithm returns one such pair in time

$$\begin{cases} \widetilde{O}\big((NM/K)^{1/3} \cdot \tau\big) & \text{if } N \leq M \leq KN^2, \\ \widetilde{O}\big((M/K)^{1/2} \cdot \tau\big) & \text{if } M \geq KN^2. \end{cases}$$

Proof. If $N \leq M \leq KN^2$ the algorithm consists of running a quantum walk over the product Johnson graph $J(N, r) \times J(M, r)$ with $r = (NM/K)^{1/3}$. This walk has spectral gap $\delta = \Omega(1/r)$ and the fraction ϵ of vertices containing both elements from at least one marked pair satisfies $\epsilon \geq \Omega\left(\frac{r^2 K}{NM}\right)$ by Lemma 2.7. Using the MNRS framework [MNRS11], the query complexity of finding one marked pair is then $O(S + \frac{1}{\sqrt{\epsilon}}(\frac{U}{\sqrt{\delta}} + C))$, where the setup cost is $S = r$, the update cost is $U = O(1)$, and the checking cost is $C = 0$. This leads to a query complexity of $O\big(r + 1/\sqrt{\epsilon\delta}\big) = O((NM/K)^{1/3})$. By a simple adaptation of the data structures described in [Amb07, Section 6.2] or [Jef14, Section 3.3.4], this can be converted to a similar upper bound on the time complexity with a multiplicative overhead of τ.

If $M \geq KN^2$, the algorithm instead stores all pairs $\{(x, f(x))\}_{x \in [N]}$ in a table – sorted according to the value of the first coordinate – and then runs the quantum search algorithm on the function $F : [M] \to \{0, 1\}$ where $F(x') = 1$ if there exists $x \in [N]$ such that $\mathcal{R}(x, y) = 1$. There are at least K marked items and F can be evaluated in time $O(\tau + \log N)$ using the sorted table. Thus, the running time is $\widetilde{O}(N \cdot \tau + (M/K)^{1/2} \cdot (\tau + \log N)) = \widetilde{O}((M/K)^{1/2})$ by Fact 2.5. □

3 Dynamic programming data structure

Here we introduce our dynamic programming data structure and show how it can be used to implement low-cost queries to the elements of the set $T_{p,k}$ defined as follows.

Definition 3.1. Let $A = \{a_1, \ldots, a_n\}$ be a multiset of n integers. For integers $p \geq 2$ and $k \in \{0, 1, \ldots, p-1\}$, define the set $T_{p,k}$ by $T_{p,k} = \{S \subseteq [n] : \Sigma_A(S) \equiv k \pmod{p}\}$, and denote the cardinality of $T_{p,k}$ by $t_{p,k} := |T_{p,k}|$.

Our main tool is the table t_p, defined below, constructed by dynamic programming. As $t_{p,k} = t_p[n, k]$, once the table is constructed, the size $t_{p,k}$ of $T_{p,k}$ can be read off the last row.

Lemma 3.2. *Let n, p be non-negative integers. Define the $(n+1) \times p$ table $t_p[i, j] = |\{S \subseteq \{1, \ldots, i\} : \Sigma(S) \equiv j \pmod{p}\}|$ where $i \in [0\,..\,n]$ and $j \in [0\,..\,p-1]$. Then, t_p can be constructed in time $O(n^2 p)$ by dynamic programming.*

Proof. To compute the elements of the table, observe that $t_p[0, 0] = 1$ and $t_p[0, j] = 0$ for $j > 0$. The remaining elements can be deduced from the relation $t_p[i, j] = t_p[i-1, j] + t_p[i-1, (j - a_i) \bmod p]$. The i^{th} row of t_p can thus be deduced from the $(i-1)^{th}$ row and a_i in time $O(np)$ (the n factor is due to the entries being exponentially large) and the computation of all rows can be completed in time $O(n^2 p)$. □

3.1 Construction of the Subset-Sum oracle

We now show how to use the table t_p to quickly query any element of $T_{p,k}$. To do so, we first define an ordering of the elements of $T_{p,k}$.

Definition 3.3. Let $\prec$ be the relation over $\mathcal{P}([n])$ defined as follows: for all $S_1, S_2 \subseteq [n]$, $S_1 \prec S_2$ if and only if $\max\{i : i \in S_1 \Delta S_2\} \in S_2$.

Lemma 3.4. *The relation $\prec$ is a strict total order.*

Proof. For every subset $S \subseteq [n]$, we define $\chi(S) = \sum_{i \in S} 2^i$. Then, $S_1 \prec S_2$ if and only if $\chi(S_1) < \chi(S_2)$. Since $<$ is a total order over the integers, so is $\prec$ over $\mathcal{P}([n])$. □

Using the above relation, we now show that the query function $f : [1\,..\,t_{p,k}] \to T_{p,k}$, defined by $f(I) = S_I$, can be computed in time $O(n^2)$.

Theorem 3.5. *Let $T_{p,k}$ be enumerated as $T_{p,k} = \{S_1, \ldots, S_{t_{p,k}}\}$ where $S_1 \prec \cdots \prec S_{t_{p,k}}$. Given any integer $I \in [1\,..\,t_{p,k}]$ and random access to the elements of the table t_p, the set S_I can be computed in time $O(n^2)$.*

Proof. Algorithm 1 gives a process that starts from $t_p[n,k]$ (i.e. the total number of subsets $S \subseteq [n]$ that sum to k modulo p) and an empty set Z, and constructs S_I by going backwards ($i = n, \ldots, 1$) through the rows of t_p. At the i-th step we examine $t_p[i-1,j]$ and decide whether to include i in Z or not. If we do include i then we examine another element in that row to decide a new value of I, and we also reset j.

Algorithm 1: Fast Subset-Sum oracle

Input: Table t_p, integers $k \in [0\,..\,p-1]$ and $I \in [1\,..\,t_{p,k}]$.
Output: The Ith subset $Z \subseteq [n]$ (according to $\prec$) such that $\Sigma(Z) \equiv k \pmod{p}$.

Set $j = k$ and $Z = \varnothing$.
for $i = n, \ldots, 1$ **do**
    if $I \leq t_p[i-1,j]$ **then**
        Do nothing.
    else
        Update $Z = Z \cup \{i\}$, $I = I - t_p[i-1,j]$ and $j = j - a_i \mod p$.
Return Z.

The algorithm consists of n iterations, each of which can be performed in time $O(n)$ assuming random access to the (exponentially large) elements of t_p, and therefore the running time is $O(n^2)$. What is left to prove is that the output of the algorithm is indeed S_I.

We first provide a high-level explanation of why the algorithm works. The total ordering defined by $\prec$ implies that $T_{p,k}$ can be written as the disjoint union of two sets, $T_{p,k} = \{S_1, \ldots, S_{t_p[n-1,k]}\} \cup \{S_{t_p[n-1,k]+1}, \ldots, S_{t_p[n,k]}\}$, where n is not contained in any S_i in the first (left) set, and is contained in every S_i of the second (right) set. Thus, we add n to the working set Z only if $I > t_p[n-1,k]$. If this is the case, S_I is the $I - t_p[n-1,k]$-th element of the right set. We note that removing n from each S_i in the right set gives the next bin defined over a smaller universe of size $n-1$, $\{S \subseteq [n-1] : \Sigma(S) \equiv k - a_n \pmod{p}\}$ that has $t_p[n-1,(k-a_n) \bmod p]$ elements. Therefore, by updating $n \leftarrow n-1$, $I \leftarrow I - t_p[n-1,k]$ and $k \leftarrow (k-a_n) \bmod p$ we can repeat the process to determine whether to add $n-1$ to the working set, and so on, until we reach the value 1.

More formally, denote by I_i, j_i and Z_i the values of the respective variables at the beginning of the ith iteration. With this notation we initially have $I_n = I, j_n = k$ and $Z_n = \varnothing$, and the final output corresponds to Z_0. We prove by backwards induction for $i = n, \ldots, 1$ the following two statements, which clearly hold for $i = n$:

1. $Z_i = S_I \cap [i+1\,..\,n]$,
2. $I_i \leq t_p[i,j_i]$ and in the enumeration of $T_p[i,j_i] := \{S_1, \ldots S_{t_p[i,j_i]}\}$ according to $\prec$ we have $S_{I_i} = S_I \cap [1\,..\,i]$.

Note that the first statement implies that the final output is $Z_0 = S_I$. To prove the inductive step for the first statement, we must show that $Z_{i-1} = S_I \cap [i\,..\,n]$. The inductive hypothesis implies that this holds exactly when $i \in Z_{i-1} \Leftrightarrow i \in S_I$. Observe that $T_p[i-1,j_i] = \{S \in T_p[i,j_i] : i \notin S\}$. Therefore the set $T_p[i,j_i]$ is the distinct union of $T_p[i-1,j_i]$ and $\mathcal{F}$ where $\mathcal{F} = \{S \in T_p[i,j_i] : i \in S\}$. By the definition of $\prec$ over $T_p[i,j_i]$, for every $X \in T_p[i-1,j_i]$ and for every $Y \in \mathcal{F}$, we have $X \prec Y$. This implies that $i \in S_\ell \Leftrightarrow t_p[i-1,j_i] < \ell$, for every $\ell \in [1\,..\,t_p[i,j_i]]$. Therefore we have the following equivalences: $i \in Z_{i-1} \Leftrightarrow t_p[i-1,j_i] < I_i \Leftrightarrow i \in S_{I_i} \Leftrightarrow i \in S_I \cap [1\,..\,i] \Leftrightarrow i \in S_I$, where the first equivalence follows from the definition of Z_{i-1} and the third equivalence from the second statement of the inductive hypothesis.

We now prove the inductive step for the second statement. Let the enumeration of $T_p[i-1, j_{i-1}]$ according to $\prec$ be $T_p[i-1, j_{i-1}] = \{S'_1, \ldots S'_{t_p[i-1,j_{i-1}]}\}$. We analyse separately the case $I_i \leq t_p[i-1, j_i]$ and the case $t_p[i-1, j_i] < I_i$.

When $I_i \leq t_p[i-1, j_i]$ then $I_{i-1} \leq t_p[i-1, j_{i-1}]$ because $I_{i-1} = I_i$ and $j_{i-1} = j_i$. Also, $S'_\ell = S_\ell$, for every $\ell \in [1 \,..\, t_p[i-1, j_{i-1}]]$. Therefore we have the following equalities $S'_{I_{i-1}} = S_{I_i} = S_I \cap [1 \,..\, i] = S_I \cap [1 \,..\, i-1]$, where the second equality is the inductive hypothesis, and the third equality holds because $i \notin S_{I_i}$ when $I_i \leq t_p[i-1, j_i]$.

When $t_p[i-1, j_i] < I_i \leq t_p[i, j_i]$ then $I_{i-1} \leq t_p[i-1, j_{i-1}]$ because $I_{i-1} = I_i - t_p[i-1, j_i]$ and $t_p[i-1, j_{i-1}] = t_p[i, j_i] - t_p[i-1, j_i]$ (here we used the definition of j_{i-1}). Also, $S'_\ell = S_{\ell + t_p[i-1, j_i]} \setminus \{i\}$, for every $\ell \in [1 \,..\, t_p[i-1, j_{i-1}]]$. Therefore we have the following equalities: $S'_{I_{i-1}} = S_{I_i} \setminus \{i\} = (S_I \cap [1 \,..\, i]) \setminus \{i\} = S_I \cap [1 \,..\, i-1]$, where again the second equality is the inductive hypothesis. $\square$

As a direct corollary, we obtain a new method for enumerating solutions to Subset-Sum.

Corollary 3.6 (Enumerating solutions to Subset-Sum via dynamic programming). *Let $A = \{a_1, \ldots, a_n\}$ and $T_{p,k} = \{S \subseteq [n] : \Sigma_A(S) \equiv k \pmod p\}$. For any $c \leq |T_{p,k}|$, it is possible to find c elements of $T_{p,k}$ in time $\widetilde{O}(p + c)$.*

Proof. By Lemma 3.2 the table $t_p[i, j]$ can be constructed in time $O(n^2 p)$. Thereafter, by Theorem 3.5 each set S_I for $I \in [1 \,..\, t_{p,k}]$ can be computed in time $O(n^2)$. $\square$

An alternative method for enumerating solutions was previously known:

Fact 3.7 (Enumerating solutions to Subset-Sum [BCJ11]). *Let $A = \{a_1, \ldots, a_n\}$ where $a_i = 2^{O(n)}$ for all i. Let $p = 2^{O(n)}$, $0 \leq k \leq p-1$, and $T_{p,k} = \{S \subseteq [n] : \Sigma_A(S) \equiv k \pmod p\}$. Then, for any $c \leq |T_{p,k}|$, it is possible to find c elements of $T_{p,k}$ in time $\widetilde{O}(2^{n/2} + c)$.*

In comparison with Fact 3.7, enumerating solutions via dynamic programming is advantageous when $p < 2^{n/2}$.

3.2 Statistics about random bins

We describe some statistics about the distribution of the sets $T_{p,k}$ (Definition 3.1) when $b \in (0,1)$ is a constant, p is a random integer in $[2^{bn} \,..\, 2^{bn+1}]$, and k is a random integer in $[0 \,..\, p-1]$. Therefore, in this section, we stress out that $T_{p,k}$ is a random bin and its cardinality $t_{p,k}$ is a random integer. We first provide an upper bound on the expectation of $t_{p,k}$.

Lemma 3.8. *The expected bin size can be upper bounded as $\mathbb{E}_{p,k}[t_{p,k}] \leq 2^{(1-b)n}$.*

Proof. The expected size of $T_{p,k}$ is at most $\mathbb{E}_{p,k}[t_{p,k}] \leq 2^{(1-b)n}$ since $\{T_{p,k} : 0 \leq k < p\}$ is a partition of $\mathcal{P}([n])$ with $p \geq 2^{bn}$. $\square$

This result is extended to an upper bound on the second moment of $t_{p,k}$, under the assumption that the input does not contain too many solution pairs. This bound is needed to analyse the complexity of the variable-time amplitude amplification algorithm.

Lemma 3.9. *Fix any integer $s \geq 0$ and any real $b \in [0,1]$. If there are at most $2^{(2-b)n}$ pairs $(S_1, S_2) \in \mathcal{P}([n])^2$ such that $\Sigma(S_1) = \Sigma(S_2) + s$, then the expected product of the sizes of two bins at distance $s \bmod p$ from each other is at most $\mathbb{E}_{p,k}[t_{p,k} t_{p,(k-s) \bmod p}] \leq \widetilde{O}(2^{2(1-b)n})$.*

Proof. The expectation of $t_{p,k}t_{p,(k-s) \bmod p}$ is equal to the expected number of pairs (S_1, S_2) such that $\Sigma(S_1)$ and $\Sigma(S_2) + s$ are congruent to k modulo p, that is $\mathbb{E}_{p,k}[t_{p,k}t_{p,(k-s) \bmod p}] = \mathbb{E}_{p,k}\big[\sum_{S_1,S_2} \mathbb{1}_{\Sigma(S_1)\equiv_p\Sigma(S_2)+s\equiv_p k}\big]$. Since k is uniformly distributed in $[0\,..\,p-1]$, this is equal to $\sum_{S_1,S_2} \mathbb{E}_p[\frac{1}{p}\mathbb{1}_{\Sigma(S_1)\equiv_p\Sigma(S_2)+s}] \leq 2^{-bn}\sum_{S_1,S_2} \Pr_p[\Sigma(S_1) \equiv_p \Sigma(S_2) + s]$, using that $p \geq 2^{bn}$. It decomposes as $2^{-bn}\big(\sum_{\Sigma(S_1)=s+\Sigma(S_2)} 1 + \sum_{\Sigma(S_1)\neq s+\Sigma(S_2)} \Pr_p[\Sigma(S_1) \equiv_p \Sigma(S_2) + s]\big)$, where the first inner term is at most $2^{(2-b)n}$ by assumption, and the second term is at most $2^{2n}n2^{-bn}$ by Fact 2.2. Thus, $\mathbb{E}[t_{p,k}t_{p,(k-s) \bmod p}] \leq O\big(2^{-bn}\big(2^{(2-b)n} + 2^{2n}n2^{-bn}\big)\big) \leq \widetilde{O}(2^{2(1-b)n})$. ☐

Finally, we provide a lower bound on the number of *distinct* subset sum values that get hashed to the random bin $T_{p,k}$.

Lemma 3.10. *Let V be any subset of the image set $\{v \in \mathbb{N} : \exists S \subseteq [n], \Sigma(S) = v\}$. Let $v_{p,k}$ denote the number of values $v \in V$ such that $v \equiv k \pmod p$. Suppose that $|V| \geq 2^{(1-\ell)n}$ for some $\ell \in [0, 1]$. Then,*

$$\begin{cases} \Pr_{p,k}[v_{p,k} \geq 2^{(1-\ell-b)n-2}] = \Omega(1/n), & \text{when } \ell \leq 1-b, \\ \Pr_{p,k}[v_{p,k} \geq 1] = \Omega\big(\min\big(1/n, 2^{(1-\ell-b)n}\big)\big), & \text{when } \ell > 1-b. \end{cases}$$

Proof. The expected size of $V_{p,k}$ is at least $\mathbb{E}_{p,k}[v_{p,k}] \geq |V|/p \geq |V|2^{-bn-1}$ since $\{V_{p,k} : 0 \leq k < p\}$ is a partition of V. Similarly to Lemma 3.9, the second moment satisfies that $\mathbb{E}_{p,k}[v_{p,k}^2] \leq O\big(2^{-bn}(|V| + |V|^2n2^{-bn})\big)$ by using Fact 2.2. If $\ell \leq 1-b$ we can further simplify this bound into $\mathbb{E}_{p,k}[v_{p,k}^2] \leq O\big(2^{-bn}|V|\big)$ since $|V| \geq 2^{(1-\ell)n}$ by assumption. Finally, the result is obtained by applying the Paley–Zygmund inequality $\Pr[v_{p,k} \geq \mathbb{E}[v_{p,k}]/2] \geq \frac{\mathbb{E}[v_{p,k}]^2}{4\mathbb{E}[v_{p,k}^2]}$ and the fact that $\Pr[v_{p,k} \geq 1] = \Pr[v_{p,k} > 0]$ since $v_{p,k}$ is an integer. ☐

4 Subset-Sum

As an illustration of the utility of the data structure introduced in Section 3, here we show how it can be used to give quantum and classical algorithms for *worst-case* instances of Subset-Sum (Problem 1) based on the representation technique, with running times $\widetilde{O}(2^{n/3})$ and $\widetilde{O}(2^{n/2})$ respectively. These algorithms therefore achieve the same complexity as the best-known algorithms for worst-case complexity based on the meet-in-the-middle principle. Both algorithms use, as a first step, a simple search procedure to handle the case where many solutions exist. Note that the tables constructed by the algorithms do not depend on the target value m.

Algorithm 2: Quantum representation technique for Subset-Sum

Input: Instance of Subset-Sum of size n and target $m \leq 2^{4n}$.
Output: A subset $S \subseteq [n]$ satisfying $\Sigma(S) = m$ if one exists, otherwise output None.

1. Run the quantum search algorithm (Fact 2.5) over the sets $S \in \mathcal{P}([n])$, where a set is marked if $\Sigma(S) = m$. Stop it and proceed to step 2 if the running time exceeds $\widetilde{O}(2^{n/3})$, otherwise output the set it found within the allotted time.
2. Choose a random prime $p \in [2^{n/3}\,..\,2^{n/3+1}]$.
3. Construct the table $t_p[i, j]$ for $i = 0, \ldots, n$ and $j = 0, \ldots, p-1$ (see Section 3).
4. Run quantum search on $T_{p,m \bmod p}$ marking the sets $S \in T_{p,m \bmod p}$ satisfying $\Sigma(S) = m$.

Theorem 4.1 (Subset-Sum, quantum)**.** *Algorithm 2 solves* Subset-Sum *in time $\widetilde{O}(2^{n/3})$ with high probability.*

Proof. If the number of solutions is at least $|\{S : \Sigma(S) = m\}| \geq 2^{n/3}$ then step 1 suffices to solve the problem with high probability since quantum search needs time $\widetilde{O}(\sqrt{2^n/2^{n/3}}) = \widetilde{O}(2^{n/3})$. Hence, let us assume that the number of solutions is at most $2^{n/3}$. The table t_p can be constructed in time $\widetilde{O}(2^{n/3})$ according to Lemma 3.2. The expected size of $T_{p,m \bmod p}$ can be bounded by $\mathbb{E}_p[t_{p,m \bmod p}] = |\{S : \Sigma(S) = m\}| + \sum_{S:\Sigma(S)\neq m} \Pr[\Sigma(S) \equiv_p m] = \widetilde{O}(2^{2n/3})$ since $\Pr[\Sigma(S) \equiv_p m] \in O(n/2^{n/3})$ by Fact 2.2. Finally, by Markov's inequality, with high probability $t_{p,m \bmod p}$ is no more than a small multiple of this expectation, and quantum search over a bin of size $\widetilde{O}(2^{2n/3})$ takes time $\widetilde{O}(2^{n/3})$ (using the fast oracle of Theorem 3.5). □

The classical algorithm is similar to the quantum one, except that it constructs a bigger table t_p to balance the cost of this construction and the cost of the classical search.

Algorithm 3: Classical representation technique for SUBSET-SUM

Input: Instance of SUBSET-SUM of size n and target $m \leq 2^{4n}$.
Output: A subset $S \subseteq [n]$ satisfying $\Sigma(S) = m$ if one exists, otherwise output None.

1 Sample $2^{n/2}$ subsets $S \subseteq [n]$ uniformly at random and check if $\Sigma(S) = m$. If no solution is found, proceed to step 2.
2 Choose a random prime $p \in [2^{n/2} \mathinner{..} 2^{n/2+1}]$.
3 Construct the table $t_p[i, j]$ for $i = 0, \ldots, n$ and $j = 0, \ldots, p-1$ (see Section 3).
4 Enumerate the elements of $T_{p,m \bmod p}$ until finding $S \in T_{p,m \bmod p}$ that satisfies $\Sigma(S) = m$.

Theorem 4.2 (SUBSET-SUM, classical)**.** *Algorithm 3 solves* SUBSET-SUM *in time $\widetilde{O}(2^{n/2})$ with high probability.*

Proof. Step 1 does not suffice to solve the problem when the number of solutions is smaller than $O(2^{n/2})$. Let us suppose that we are in this case. From Lemma 3.2, the $(n+1) \times p$ table t_p can be constructed in time $\widetilde{O}(2^{n/2})$, after which each query to $T_{p,m \bmod p} = \{S \subseteq [n] : \Sigma(S) \equiv_p m\}$ can be made in time $O(n^2)$. By linearity of expectation, the expected size of $T_{p,m \bmod p}$ can be bounded by $\mathbb{E}_p[t_{p,m \bmod p}] = |\{S : \Sigma(S) = m\}| + \sum_{S:\Sigma(S)\neq m} \Pr[\Sigma(S) \equiv_p m] = \widetilde{O}(2^{n/2})$ since, by Fact 2.2, $\Pr[\Sigma(S) \equiv_p m] \in O(n/2^{n/2})$ for each of the (at most 2^n) sets S for which $\Sigma(S) \neq m$. By Markov's inequality, with high probability $t_{p,m \bmod p}$ is no more than a small multiple of this expectation and thus can be enumerated in time $\widetilde{O}(2^{n/2})$ using Theorem 3.5. □

5 SHIFTED-SUMS

In this section we present the two quantum algorithms for solving SHIFTED-SUMS (Problem 6). The running time of both algorithms – expressed in Theorems 5.5 and 5.7 – are functions of the size of a *maximum solution* of the input. This notion plays a central role in our algorithms and is defined next.

Definition 5.1 (Maximum solution)**.** We say that two disjoint subsets $S_1, S_2 \subseteq [n]$ that form a solution to an instance of SHIFTED-SUMS are a *maximum solution* if the size $|S_1| + |S_2| = \ell n$ is the largest among all such solutions. We call $\ell \in (0, 1)$ the maximum solution *ratio.*

By choosing the faster of these two algorithms for each $\ell \in \{1/n, 2/n, \ldots, (n-1)/n\}$ until a solution has been found (or it can be concluded that no solution exists), we obtain an overall quantum algorithm for SHIFTED-SUMS with the following performance:

Theorem 5.2 (Shifted-Sums, quantum)**.** *There is a quantum algorithm that, given an instance of* Shifted-Sums *with maximum solution ratio $\ell \in (0,1)$, outputs a solution with at least inverse polynomial probability in time $\widetilde{O}(2^{\gamma(\ell)n})$ where*

$$\gamma(\ell) = \begin{cases} (1+\ell)/4 & \text{if } \ell_1 \le \ell \le 3/5, & \text{(Theorem 5.5)} \\ \ell/2 + 1/10 & \text{if } 3/5 < \ell < \ell_2, & \text{(Theorem 5.5)} \\ (h(\ell)+\ell)/3 & \text{otherwise} & \text{(Theorem 5.7)} \end{cases}$$

and $\ell_1 \approx 0.190$ and $\ell_2 \approx 0.809$ are solutions to the equations $(h(\ell)+\ell)/3 = (1+\ell)/4$ and $(h(\ell)+\ell)/3 = \ell/2 + 1/10$ respectively. In particular, the worst-case complexity is $O(2^{0.504n})$.

Since a potential solution can be verified in polynomial time in n, in what follows we describe our algorithms on yes instances with maximum solution ratio ℓ. As presented, the algorithms find a solution with inverse polynomial probability in n, which can be amplified to constant probability in polynomial time. Recall the classical algorithm of [MNPW19] for Equal-Sums is based on the concept of a *minimum* solution (Definition B.1), rather than a *maximum* solution. In Appendix B, we present an analogous quantum algorithm for Equal-Sums whose complexity is expressed in terms of the minimum solution ratio ℓ' (we do not know how to extend this result to Shifted-Sums). While this does not change the algorithmic complexity in the worst case, for a given instance of Equal-Sums the quantity ℓ' may be smaller than ℓ.

5.1 Representation technique algorithm

Our representation-technique-based algorithm is given in Algorithm 4, and uses the dynamic programming table of Section 3. Before constructing that table, we first check whether the input contains many solution pairs (in which case a simple quantum search is sufficient). Depending on the value of the maximum solution ratio ℓ, we may also need to apply variable-time amplitude amplification (Fact 2.6) on top of quantum pair finding (Theorem 2.8).

Algorithm 4: Quantum representation technique for Shifted-Sums

Input: Instance (a, s) of Shifted-Sums with $\sum_{i=1}^{n} a_i < 2^{4n}$ and maximum solution ratio ℓ.

Output: Two subsets $S_1, S_2 \subseteq [n]$.

1 Set $b = (1+\ell)/4$ if $\ell \le 3/5$ and $b = 2/5$ if $\ell > 3/5$.

2 Run the quantum search algorithm (Fact 2.5) over the set of pairs $(S_1, S_2) \in \mathcal{P}([n])^2$, where a pair is marked if $\Sigma(S_1) = \Sigma(S_2) + s$ and $S_1 \neq S_2$. Stop it and proceed to step 3 if the running time exceeds $\widetilde{O}(2^{bn/2})$, otherwise output the pair it found within the allotted time.

3 If $\ell > 3/5$ then run variable-time amplitude amplification (Fact 2.6) on steps 4–6, otherwise run them once:

4 Choose a random prime $p \in [2^{bn} \mathinner{..} 2^{bn+1}]$ and a random integer $k \in [0 \mathinner{..} p-1]$.

5 Construct the table $t_p[i,j]$ for $i = 0, \ldots, n$ and $j = 0, \ldots, p-1$ (see Section 3).

6 Run the quantum pair finding algorithm (Theorem 2.8) to find if there exist two sets $S_1 \in T_{p,k}$ and $S_2 \in T_{p,(k-s) \bmod p}$ such that $\Sigma(S_1) = \Sigma(S_2) + s$ and $S_1 \neq S_2$. If so, output the pair (S_1, S_2) it found.

Note that 'run variable-time amplitude amplification on steps 4–6' means that one should apply the procedure implicit in Fact 2.6 to the algorithm $\mathcal{A}$ defined by the following process (i) Create a uniform superposition over all primes $p \in [2^{bn} \mathinner{..} 2^{bn+1}]$ and, for each p, all $k \in [0 \mathinner{..} p-1]$. (ii) For each p, coherently construct the table t_p. (iii) Run quantum pair finding coherently on each pair of sets $T_{p,k}, T_{p,(k-s) \bmod p}$, marking the (p, k) tuple if a pair is found.

The analysis of the above algorithm relies on the random bin statistics presented in Section 3.2. We first define the *collision values set* which contains the values of all the possible solution pairs.

Definition 5.3 (Collision Values set)**.** Given an instance (a, s) to the Shifted-Sums problem, the *collision values set* is the set $V = \{v \in \mathbb{N} : \exists S_1 \neq S_2, v = \Sigma(S_1) = \Sigma(S_2) + s\}$.

We show that the collision values set V is of size at least $2^{(1-\ell)n}$ when the maximum solution ratio is ℓ. Thus, by Lemma 3.10, we can lower bound the number of values in V that get hashed to a random bin $T_{p,k}$.

Lemma 5.4. *If the maximum solution ratio is ℓ then $|V| \geq 2^{(1-\ell)n}$.*

Proof. Let $S_1, S_2 \subseteq \{1, \ldots, n\}$ be a maximum solution of size $|S_1| + |S_2| = \ell n$. Then for any $S \subseteq [n] \setminus (S_1 \cup S_2)$ the sets $S_1 \cup S$ and $S_2 \cup S$ form a solution, and for $S \neq S'$, the values $\Sigma(S_1 \cup S)$ and $\Sigma(S_1 \cup S')$ must be distinct. If this were not the case then $S_1 \cup (S \setminus S')$ and $S_2 \cup (S' \setminus S)$ would form a disjoint solution of size larger than ℓ. □

We finally analyse Algorithm 4 in the next theorem.

Theorem 5.5 (Shifted-Sums, representation)**.** *Given an instance of* Shifted-Sums *with $\sum_{i=1}^{n} a_i < 2^{4n}$ and maximum solution ratio $\ell \in (0, 1)$,* Algorithm 4 *finds a solution with inverse polynomial probability in time $\widetilde{O}(2^{(1+\ell)n/4})$ if $\ell \leq 3/5$, and $\widetilde{O}(2^{(\ell/2+1/10)n})$ if $\ell > 3/5$.*

Proof. Step 2 of Algorithm 4 handles the case where the total number of solution pairs exceeds $2^{(2-b)n}$. In this situation, the quantum search algorithm can find a solution pair in time $\widetilde{O}(\sqrt{2^{2n}/2^{(2-b)n}}) = \widetilde{O}(2^{bn/2})$, which is smaller than the complexity stated in Theorem 5.5.

Analysis when $\ell \leq 3/5$. In this case the algorithm executes steps 4–6 only once. From Lemma 3.2, the table t_p can be constructed in time $\widetilde{O}(2^{bn})$, after which each query to the elements of $T_{p,k}$ can be performed in time $O(n^2)$ (Theorem 3.5). By Lemma 3.10, the number of disjoint solution pairs contained in $T_{p,k} \times T_{p,(k-s) \bmod p}$ is at least $v_{p,k} \geq 2^{(1-\ell-b)n-2}$ with probability $\Omega(1/n)$. By Lemma 3.8 and Markov's inequality, the sizes of $T_{p,k}$ and $T_{p,(k-s) \bmod p}$ are at most $t_{p,k}, t_{p,(k-s) \bmod p} \leq n^2 2^{(1-b)n}$ with probability at least $1 - 1/n^2$. Thus, with probability $\Omega(1/n)$ we can assume that both of these events occur. If this is the case, then the time to execute step 6 of the algorithm is $\widetilde{O}\Big(\big(t_{p,k} t_{p,(k-s) \bmod p}/v_{p,k}\big)^{1/3}\Big) = \widetilde{O}(2^{(1+\ell-b)n/3})$ since the first complexity given in Theorem 2.8 is the largest one for our choice of parameters. This is at most $\widetilde{O}(2^{(1+\ell)n/4})$ when $b = (1+\ell)/4$.

Analysis when $\ell > 3/5$. We assume that the total number of solution pairs is at most $2^{(2-b)n}$ (otherwise we would have found a collision at step 2 with high probability). Given p and k, the base algorithm (steps 4–6) succeeds if there is a solution in $T_{p,k} \times T_{p,(k-s) \bmod p}$, i.e. $v_{p,k} \geq 1$. Therefore by Lemmas 5.4 and 3.10, we have for its success probability $\rho = \Omega(\min(1/n, 2^{(1-\ell-b)n}))$. We claim that $\mathbb{E}[\tau^2] = \widetilde{O}(2^{2bn})$ where τ is the stopping time of the base algorithm. Constructing the table t_p takes time $\widetilde{O}(p)$, and by summing the two complexities given in Theorem 2.8 the quantum pair finding algorithm takes time at most $\widetilde{O}\big((t_{p,k} t_{p,(k-s) \bmod p})^{1/3} + \sqrt{\max(t_{p,k}, t_{p,(k-s) \bmod p})}\big)$. Therefore we have

$$
\begin{aligned}
\mathbb{E}[\tau^2] &= \widetilde{O}\bigg(\mathbb{E}_{p,k}\bigg[\Big(p + \big(t_{p,k} t_{p,(k-s) \bmod p}\big)^{1/3} + \sqrt{\max(t_{p,k}, t_{p,(k-s) \bmod p})}\Big)^2\bigg]\bigg) \\
&\leq \widetilde{O}\Big(\mathbb{E}_{p,k}\big[p^2\big] + \mathbb{E}_{p,k}\Big[t_{p,k}^{2/3} t_{p,(k-s) \bmod p}^{2/3}\Big] + \mathbb{E}_{p,k}\big[t_{p,k}\big]\Big) \\
&\leq \widetilde{O}\Big(\mathbb{E}_{p,k}\big[p^2\big] + \mathbb{E}_{p,k}\Big[t_{p,k} t_{p,(k-s) \bmod p}\Big]^{2/3} + \mathbb{E}_{p,k}\big[t_{p,k}\big]\Big) \\
&\leq \widetilde{O}(2^{2bn}) + \widetilde{O}(2^{4(1-b)n/3}) + 2^{(1-b)n},
\end{aligned}
$$

where the second inequality uses that the moment function is non-decreasing and the last inequality uses Lemmas 3.8 and 3.9. Since $b = 2/5$ we obtain that $\mathbb{E}[\tau^2] \leq \widetilde{O}(2^{2bn})$. Finally, by Fact 2.6, the overall time of steps 3–6 is $\widetilde{O}(\sqrt{\mathbb{E}[\tau^2]/\rho}) = \widetilde{O}(2^{bn}/2^{(1-\ell-b)n/2}) = \widetilde{O}(2^{(\frac{\ell}{2}+\frac{1}{10})n})$. □

5.2 Meet-in-the-middle algorithm

Our second algorithm uses the standard meet-in-the-middle technique combined with quantum search to solve the Shifted-Sums problem. We first state a lemma that if we randomly partition the input into two sets of relative sizes 1:2, then with at least inverse polynomial probability a maximum solution will be distributed in the same proportion between the two sets.

Lemma 5.6. *Let S_1, S_2 be a maximum solution of ratio ℓ. Then with at least inverse polynomial probability the random partition $X_1 \cup X_2$ satisfies $|(S_1 \cup S_2) \cap X_1| = \ell n/3$, $|(S_1 \cup S_2) \cap X_2| = 2\ell n/3$.*

Proof. There are $\binom{n}{n/3}$ ways to partition $[n]$ into two subsets X_1 and X_2 of respective sizes $n/3$ and $2n/3$. Of these, there are $\binom{\ell n}{\ell n/3} \cdot \binom{n-\ell n}{\frac{n-\ell n}{3}}$ partitions such that $|(S_1 \cup S_2) \cap X_1| = \ell n/3, |(S_1 \cup S_2) \cap X_2| = 2\ell n/3$. The probability that $|(S_1 \cup S_2) \cap X_1| = \ell n/3, |(S_1 \cup S_2) \cap X_2| = 2\ell n/3$ is thus $\frac{\binom{\ell n}{\ell n/3} \cdot \binom{n-\ell n}{\frac{n-\ell n}{3}}}{\binom{n}{n/3}}$. Fact 2.1 gives that this quantity is at least $\Omega\big(n^{-1/2}\big)$. □

We use the above result in the design of Algorithm 5, which is analysed in the next theorem. We observe that the obtained time complexity is at most $\widetilde{O}(3^{n/3})$ and is maximized at $\ell = 2/3$.

Algorithm 5: Quantum meet-in-the-middle technique for Shifted-Sums

Input: Instance (a, s) of Shifted-Sums with maximum solution ratio ℓ.
Output: Two subsets $S_1, S_2 \subseteq [n]$.

1. Randomly split $[n]$ into disjoint subsets $X_1 \cup X_2$ such that $|X_1| = n/3, |X_2| = 2n/3$.
2. Classically compute and sort $V_1 = \{\Sigma(S_{11}) - \Sigma(S_{21}) : S_{11}, S_{21} \subseteq X_1 \text{ and } S_{11} \cap S_{21} = \emptyset \text{ and } |S_{11}| + |S_{21}| = \ell n/3\}$.
3. Apply quantum search (Fact 2.5) over the set $V_2 = \big\{\Sigma(S_{12}) - \Sigma(S_{22}) : S_{12}, S_{22} \subseteq X_2 \text{ and } S_{12} \cap S_{22} = \emptyset \text{ and } |S_{12}| + |S_{22}| = 2\ell n/3\big\}$, where an element $v_2 \in V_2$ is marked if there exists $v_1 \in V_1$ such that $v_1 + v_2 = s$. For a marked v_2, output $S_1 = S_{11} \cup S_{12}$ and $S_2 = S_{21} \cup S_{22}$.

Theorem 5.7 (Shifted-Sums, meet-in-the-middle)**.** *Given an instance of* Shifted-Sums *with maximum solution ratio $\ell \in (0, 1)$,* Algorithm 5 *finds a solution with at least inverse polynomial probability in time $\widetilde{O}(2^{n(h(\ell)+\ell)/3})$.*

Proof. There are $\binom{n/3}{\ell n/3} 2^{\ell n/3}$ different ways to select two sets $S_{11}, S_{21} \subseteq X_1$ such that $S_{11} \cap S_{21} = \emptyset$, $|S_{11}| + |S_{21}| = \ell n/3$. Computing and sorting V_1 thus take time $\widetilde{O}(\binom{n/3}{\ell n/3} 2^{\ell n/3})$. In the next step of the algorithm, quantum search is performed over all $\binom{2n/3}{2\ell n/3} 2^{2\ell n/3}$ sets $S_{12}, S_{22} \subseteq X_2$ such that $S_{12} \cap S_{22} = \emptyset$, $|S_{12}| + |S_{22}| = 2\ell n/3$. We mark an element $v_2 \in V_2$ if there exists $v_1 \in V_1$ such that $v_1 + v_2 = s$. Since V_1 is sorted this check can be done in time $\mathrm{polylog}(|V_1|)$. The total time required is therefore $\widetilde{O}\Big(\binom{n/3}{\ell n/3} 2^{\ell n/3} + \sqrt{\binom{2n/3}{2\ell n/3} 2^{2\ell n/3}}\Big) = \widetilde{O}\Big(2^{\ell n/3}\Big(\binom{n/3}{\ell n/3} + \sqrt{\binom{2n/3}{2\ell n/3}}\Big)\Big) = \widetilde{O}(2^{\frac{n}{3}(h(\ell)+\ell)})$. By Lemma 5.6, when the instance has a maximum solution of size ℓn, the set V_2 has a marked element with at least inverse polynomial probability, and in that case a solution is found. □

6 Pigeonhole variants of Equal-Sums

6.1 Pigeonhole Equal-Sums

We give classical and quantum algorithms for Pigeonhole Equal-Sums (Problem 8), based on dynamic programming and which run in time $\widetilde{O}(2^{n/2})$ and $\widetilde{O}(2^{2n/5})$, respectively. In contrast with our quantum algorithm for Shifted-Sums that made use of a random prime modulus, in the case of Pigeonhole Equal-Sums we can deterministically choose a modulus p, and the pigeonhole principle guarantees a collision in at least one bin.

Lemma 6.1. *There is a classical deterministic algorithm such that, given an instance of* Pigeonhole Equal-Sums *and a modulus p that divides 2^n, it finds in time $\widetilde{O}(p)$ an integer k such that there exist two distinct subsets S_1, S_2 with $\Sigma(S_1) \equiv \Sigma(S_2) \equiv k \pmod p$.*

Proof. Denote by $\overline{0}, \overline{1}, \ldots \overline{p-1}$ the congruence classes modulo p. Each of these classes contains exactly $2^n/p$ numbers between 0 and $2^n - 2$, except the last class $\overline{p-1}$ that has only $2^n/p - 1$ numbers. Since all 2^n subsets $S \subseteq [n]$ have a sum $\Sigma(S)$ between 0 and $2^n - 2$ there are two possible cases:

- either there is some class $\overline{k}$ such that $\Sigma(S) \in \overline{k}$ for strictly more than $2^n/p$ subsets S,
- or there are $2^n/p$ subsets S such that $\Sigma(S) \in \overline{p-1}$.

Denote by $\overline{k}$ a class that verifies one of these two points. By definition, there are *strictly more* subsets S such that $\Sigma(S) \in \overline{k}$ than the number of elements between 0 and $2^n - 2$ that belong to $\overline{k}$. However, for all $S \subseteq [n]$, we have $\Sigma(S) \leq 2^n - 2$. Thus, there must be two subsets $S_1 \neq S_2$ such that $\Sigma(S_1), \Sigma(S_2) \in \overline{k}$ and $\Sigma(S_1) = \Sigma(S_2)$.

From Lemma 3.2, the table $t_p[i,j] = |\{S \subseteq \{1,\ldots,i\} : \Sigma(S) \equiv j \pmod p\}|$ can be constructed in time $\widetilde{O}(p)$. From the table, we can read off a value k that satisfies the above condition. □

Theorem 6.2 (Pigeonhole Equal-Sums, classical)**.** *There is a classical deterministic algorithm for the* Pigeonhole Equal-Sums *problem that runs in time $\widetilde{O}(2^{n/2})$.*

Proof. Choose $p = 2^{n/2}$. By Lemma 6.1, in time $\widetilde{O}(2^{n/2})$ we can find k such that there exist $S_1 \neq S_2$ satisfying $\Sigma(S_1) \equiv \Sigma(S_2) \equiv k \pmod{2^{n/2}}$. Once we know a bin that contains a collision, by Corollary 3.6, we can enumerate in time $\widetilde{O}(2^{n/2})$ a sufficient number of subsets in that bin to locate a collision. □

Theorem 6.3 (Pigeonhole Equal-Sums, quantum)**.** *There is a quantum algorithm for the* Pigeonhole Equal-Sums *problem that runs in time $\widetilde{O}(2^{2n/5})$.*

Proof. We set $p = 2^{2n/5}$ and, by Lemma 6.1, in time $\widetilde{O}(2^{2n/5})$ we can identify k such that there exist $S_1 \neq S_2$ satisfying $\Sigma(S_1) \equiv \Sigma(S_2) \equiv k \pmod{2^{2n/5}}$. By Theorem 3.5, each query to $T_{p,k} = \{S \subseteq [n] : \Sigma(S) \equiv k \pmod p\}$ can be made in time $O(n^2)$. We use Ambainis' element distinctness algorithm [Amb07] on these elements to find a collision. We do not want to run it on an unnecessarily large set. Therefore, if $t_{p,k} > 2^{3n/5+1}$ then we run it only on the first $2^{3n/5+1}$ elements of $T_{p,k}$, according to the ordering defined by $\prec$. A collision is then found in time $\widetilde{O}\big((2^{3n/5})^{2/3}\big) = \widetilde{O}(2^{2n/5})$. The overall running time of the algorithm is thus $\widetilde{O}(2^{2n/5})$. □

6.2 Pigeonhole Modular Equal-Sums

Here we give a classical representation-technique-based algorithm for Pigeonhole Modular Equal-Sums (Problem 9). Our approach consists of defining the bins based on the *quotient* in the division by p instead of the remainder, that is $T_{p,k} = \big\{S \subseteq \{1,\ldots,n\} : \lfloor(\sum_{i \in S} a_i \bmod q)/p\rfloor = k\big\}$. We show that computing the cardinality of $T_{p,k}$ and enumerating its elements can be done with the help of yet another table for the bins $T'_{p,k} = \big\{S \subseteq \{1,\ldots,n\} : (\sum_{i \in S}\lfloor a_i/p\rfloor) \equiv k \pmod{\lfloor q/p\rfloor}\big\}$. This last table can be constructed with a similar dynamic programming technique as before. We do not know how to extend this approach to the more general Equal-Sums or Shifted-Sums problems due to the lack of good statistics on how a random bin $T_{p,k}$ behaves in this case. We also have no quantum speed-up for Pigeonhole Modular Equal-Sums due to a bottleneck when going from $T'_{p,k}$ to $T_{p,k}$ that we cannot seemingly reduce with quantum techniques.

Theorem 6.4 (Pigeonhole Modular Equal-Sums, classical)**.** *There is a classical deterministic algorithm for the* Pigeonhole Modular Equal-Sums *problem that runs in time* $\widetilde{O}(2^{n/2})$*.*

Proof. Without loss of generality we suppose that for every input a_i, the inequality $a_i < q$ holds, where $q \leq 2^n - 1$ is the modulus in the input. For such a modulus there exists a unique couple (q_1, q_2) with $0 \leq q_1, q_2 < 2^{n/2}$ such that $q = q_1 2^{n/2} + q_2$. We define the one-dimensional array $B[j]$ for $0 \leq j \leq q_1 - 1$ by

$$B[j] = \left\{ S \subseteq \{1, \ldots, n\} : \left\lfloor \frac{\Sigma(S) \bmod q}{2^{n/2}} \right\rfloor = j \right\}.$$

We denote the cardinality of $B[j]$ by $b[j]$, and we define $\beta[j] = \left| \left\{ 0 \leq k \leq q - 1 : \left\lfloor \frac{k}{2^{n/2}} \right\rfloor = j \right\} \right|$. Observe that $\beta[j] \leq 2^{n/2}$, for all j, and that

$$\sum_{j=0}^{q_1-1} \beta_j = q < 2^n = \sum_{j=0}^{q_1-1} b[j].$$

Therefore there exists j such that $\beta[j] < b[j]$, and we will call such an index *marked*. The algorithm will identify a marked j, and then will find $\beta[j] + 1$ different sets in $B[j]$. We will show that this can be done in time $\widetilde{O}(2^{n/2})$, and by the pigeonhole principle there are two sets $S_1, S_2 \subseteq \{1, \ldots, n\}$ among them such that $\Sigma(S_1) \equiv \Sigma(S_2) \pmod{q}$.

Computing directly the values in the array b is not easy, we will do that with the help of another one-dimensional array C. For $1 \leq k \leq n$, we set $a'_k = \left\lfloor \frac{a_k}{2^{n/2}} \right\rfloor$ and $A' = \{a'_1, \ldots, a'_n\}$. Then we define

$$C[j] = \{ S \subseteq \{1, \ldots, n\} : \Sigma_{A'}(S) \equiv j \pmod{q_1} \},$$

for $0 \leq j \leq q_1 - 1$, and we set $c[j] = |C[j]|$. By Lemma 3.2 the full array c can be computed in time $\widetilde{O}(2^{n/2})$, and by Theorem 3.5, for every $0 \leq j \leq q_1 - 1$, the entry $C[j]$ can be enumerated in $O(n^2)$ time per set.

The arrays B and C are of course not identical, but the following lemma shows that any set in $B[j]$ must be contained in C at an index close to j. For any $0 \leq i, j \leq q_1 - 1$, we define

$$B[i\,..\,j] = \begin{cases} \cup_{k=i}^{j} B[k] & \text{if } i \leq j \\ \cup_{k=i}^{q_1-1} B[k] \bigcup \cup_{k=0}^{j} B[k] & \text{otherwise,} \end{cases}$$

and $C[i\,..\,j]$ is defined analogously. We denote their respective cardinalities as $b[i\,..\,j] = |B[i\,..\,j]|$ and $c[i\,..\,j] = |C[i\,..\,j]|$. Finally, for $0 \leq i \leq j \leq q_1$, we set $\beta[i\,..\,j] = \sum_{k=i}^{j} \beta[k]$.

Lemma 6.5. *For every* $0 \leq j \leq q_1 - 1$*, the inclusion* $B[j] \subseteq C[(j - n + 1) \bmod q_1\,..\,(j + n - 1) \bmod q_1]$ *holds.*

Proof. Let $S \in B[j]$. Then by definition $\Sigma(S) \bmod q = j \cdot 2^{n/2} + j'$, for some $0 \leq j' \leq 2^{n/2} - 1$. Consequently $\Sigma(S) = k_1 q + j \cdot 2^{n/2} + j'$, where $0 \leq k_1 \leq n - 1$ because $a_i < q$ for every i, and therefore $\Sigma(S) < nq$. This implies that

$$\begin{aligned} \left\lfloor \frac{\Sigma(S)}{2^{n/2}} \right\rfloor &= j + \left\lfloor \frac{k_1 q + j'}{2^{n/2}} \right\rfloor \\ &= j + \left\lfloor \frac{k_1(q_1 2^{n/2} + q_2) + j'}{2^{n/2}} \right\rfloor \\ &= j + k_1 q_1 + \left\lfloor \frac{k_1 q_2 + j'}{2^{n/2}} \right\rfloor \\ &= j + k_1 q_1 + k_2, \end{aligned}$$

where $0 \le k_2 \le n-1$. Therefore

$$\Sigma_{A'}(S) = j + k_1 q_1 + k_2 - k_3,$$

where $0 \le k_3 \le n-1$ since the set S contains at most n elements. We can thus conclude that

$$\Sigma_{A'}(S) \equiv j + k_4 \pmod{q_1},$$

where for $k_4 = k_2 - k_3$ we have $-n+1 \le k_4 \le n-1$. □

Corollary 6.6. *Let $0 \le i, j \le q_1 - 1$ with $2n \le j - i \le (q_1-1)/2$. Then $C[i+n-1\,..\,j-n+1] \subseteq B[i\,..\,j] \subseteq C[(i-n+1) \bmod q_1\,..\,(j+n-1) \bmod q_1]$.*

Proof. The second inclusion follows immediately from Lemma 6.5. The lemma also implies $B[j+1 \bmod q_1\,..\,i-1 \bmod q_1] \subseteq C[(j-n+2)\,..\,(i+n-2)]$. Taking the complement of the set on each side gives the first inclusion. □

Corollary 6.7. *Let $0 \le i \le q_1 - 1$. Then there exists $k \in \{i-n, i+n\}$ such that $C[i] \subseteq B[(k-2n+1)\,..\,(k+2n-1)]$.*

Proof. We have either $i \ge 3n-1$ or $i \le q_1 - 3n$. If only the first case is true choose $k = i - n$, if only the second case is true choose $k = i + n$, if both cases are true choose arbitrarily. Obviously $C[i] \subseteq C[(k-n)\,..\,(k+n)]$, therefore Corollary 6.6 implies the statement. □

Corollary 6.8. *Let $0 \le i \le q_1 - 1$. In time $\widetilde{O}(2^{n/2})$ we can either enumerate $C[i]$ or we can find a marked index.*

Proof. If $c[i] \le (4n-1)2^{n/2}$ then by Theorem 3.5 we can enumerate $C[i]$. Otherwise, by Corollary 6.7, we test each $S \in C[i]$ to see which set $B[j]$, for $k - 2n + 1 \le j \le k + 2n - 1$, it belongs to until an index j satisfying $b[j] > 2^{n/2}$ is identified. □

We now describe the procedure to find a marked index. It is essentially a dichotomic search over shorter and shorter intervals $[i\,..\,j] = \{i, \ldots, j\}$, with the invariant property $\beta[i\,..\,j] < b[i\,..\,j]$, and where in every step we halve the size of $j - i$. Initially, we set $i = 0$ and $j = q_1 - 1$, and we stop the process when $2n \le j - i < 4n$. Clearly the number of iterations is less than $n/2$. We now describe one iteration. Let us suppose that our current interval is $[i\,..\,j]$, and let $m = (j-i)/2$, rounding it arbitrarily, if necessary. We will compute $b[i\,..\,m]$ and $b[m+1\,..\,j]$ and keep one of the two intervals for which the invariant property holds.

We claim that for some fixed indices i, j, in time $\widetilde{O}(2^{n/2})$ we can either compute $b[i\,..\,j]$ or we find a marked index. From Corollary 6.6 it follows that

$$\begin{aligned} b[i\,..\,j] &= c[i+n-1\,..\,j-n+1] \\ &\quad + \Big| B[i\,..\,j] \cap \big(C[(i-n+1) \bmod q_1\,..\,i+n-2] \cup C[(j-n+2)\,..\,(j+n-1) \bmod q_1] \big) \Big|. \end{aligned}$$

The first term $c[i+n-1\,..\,j-n+1]$ is computed in time $\widetilde{O}(q_1)$ by adding the corresponding entries in the array c. For the second term, since there at most $4n$ entries of C involved in it, we can either enumerate all the elements they contain in time $\widetilde{O}(2^{n/2})$ by Corollary 6.8 and check for each element if it belongs to $B[i\,..\,j]$ (hence we can compute $b[i\,..\,j]$), or we can find a marked index.

Unless we already found a marked index during the process, the dichotomic search stops with less than $4n$ candidate indices out of which at least one is marked. Therefore the last thing to show is that given a marked index j, how do we find a solution in time $\widetilde{O}(2^{n/2})$. From Lemma 6.5 and Corollary 6.6 we know that

$$B[j] \subseteq C[(j-n+1) \bmod q_1\,..\,(j+n-1) \bmod q_1] \subseteq B[j-2n+2 \bmod q_1\,..\,j+2n+2 \bmod q_1].$$

We start enumerating $\mathcal{C} = C[(j - n + 1) \bmod q_1 \,..\, (j + n - 1) \bmod q_1]$ until one of the following two things happens. If $|\mathcal{C}| \leq (4n-3)2^{n/2}$ then we fully enumerate $\mathcal{C}$ and therefore can also fully enumerate $B[j]$, and find a solution there. Otherwise, we stop after having enumerated $(4n-3)2^{n/2} + 1$ elements of $\mathcal{C}$, and for each of them we determine the index where they belong in $B[j - 2n + 2 \bmod q_1 \,..\, j + 2n + 2 \bmod q_1]$. There will be an index where we have found more than $2^{n/2}$ subsets and therefore also a solution. □

Remark 6.9. *As an anonymous referee has pointed out,* Pigeonhole Modular Equal-Sums *can also be solved in the same running time as above based on a meet-in-the-middle approach: (i) Create a list L_1 of all subsets $S \subseteq \{1, \ldots, n/2\}$ sorted in non-decreasing order of $\Sigma(S) \bmod q$, and another similar list L_2 for subsets $S \subseteq \{n/2+1, \ldots, n\}$. (ii) By performing $O(2^{n/2})$ binary searches through L_1 (one for each element of L_2), one can compute $|\{S \subseteq [n] : \Sigma(S) \bmod q \leq x\}|$ (for any x), and hence compute $|\{S \subseteq [n] : \Sigma(S) \bmod q \in [l, r]\}|$. (iii) Starting from the interval $[0, q-1]$, for each interval $[l, r]$ and $m = (l+r)/2$ recursively identify whether $[l, m]$ or $[m, r]$ contains more subsets than pigeonholes. The problem is solved in* $\operatorname{poly}(n)$ *recursions.*

7 Acknowledgements

This work has been supported by the European Union's H2020 Programme under grant agreement number ERC-669891. Research at CQT is funded by the National Research Foundation, the Prime Minister's Office, and the Ministry of Education, Singapore under the Research Centres of Excellence programme's research grant R-710-000-012-135. JA thanks Shengyu Zhang for helpful discussions during the course of this work.

References

[ABB+21] K. Axiotis, A. Backurs, K. Bringmann, C. Jin, V. Nakos, C. Tzamos, and H. Wu. "Fast and Simple Modular Subset Sum". In: *Proceedings of the 4th Symposium on Simplicity in Algorithms (SOSA)*. 2021, pp. 57–67 (cit. on p. 2).

[Abb19] A. Abboud. "Fine-Grained Reductions and Quantum Speedups for Dynamic Programming". In: *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*. 2019, 8:1–8:13 (cit. on p. 8).

[ABHS19] A. Abboud, K. Bringmann, D. Hermelin, and D. Shabtay. "SETH-Based Lower Bounds for Subset Sum and Bicriteria Path". In: *Proceedings of the 30th Symposium on Discrete Algorithms (SODA)*. 2019, pp. 41–57 (cit. on p. 2).

[ABI+19] A. Ambainis, K. Balodis, J. Iraids, M. Kokainis, K. Prusis, and J. Vihrovs. "Quantum Speedups for Exponential-Time Dynamic Programming Algorithms". In: *Proceedings of the 30th Symposium on Discrete Algorithms (SODA)*. 2019, pp. 1783–1793 (cit. on pp. 8, 9).

[ABJ+19] K. Axiotis, A. Backurs, C. Jin, C. Tzamos, and H. Wu. "Fast Modular Subset Sum using Linear Sketching". In: *Proceedings of the 30th Symposium on Discrete Algorithms (SODA)*. 2019, pp. 58–69 (cit. on p. 2).

[AKKN16] P. Austrin, P. Kaski, M. Koivisto, and J. Nederlof. "Dense Subset Sum May Be the Hardest". In: *Proceedings of the 33rd Symposium on Theoretical Aspects of Computer Science (STACS)*. 2016, 13:1–13:14 (cit. on pp. 4, 5).

[Amb07] A. Ambainis. "Quantum Walk Algorithm for Element Distinctness". In: *SIAM Journal on Computing* 37.1 (2007), pp. 210–239 (cit. on pp. 5, 6, 9, 11, 12, 20).

[Amb12] A. Ambainis. "Variable Time Amplitude Amplification and Quantum Algorithms for Linear Algebra Problems". In: *Proceedings of the 29th Symposium on Theoretical Aspects of Computer Science (STACS)*. 2012, pp. 636–647 (cit. on pp. 6, 11).

[BBHT98] M. Boyer, G. Brassard, P. Høyer, and A. Tapp. "Tight Bounds on Quantum Searching". In: *Fortschritte der Physik* 46.4-5 (1998), pp. 493–505 (cit. on p. 10).

[BBSS20] X. Bonnetain, R. Bricout, A. Schrottenloher, and Y. Shen. "Improved Classical and Quantum Algorithms for Subset-Sum". In: *Proceedings of the 26th International Conference on the Theory and Applications of Cryptology and Information Security (ASIACRYPT)*. 2020, pp. 633–666 (cit. on p. 8).

[BCJ11] A. Becker, J.-S. Coron, and A. Joux. "Improved Generic Algorithms for Hard Knapsacks". In: *Proceedings of the 30th International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*. 2011, pp. 364–385 (cit. on pp. 8, 14).

[BJLM13] D. J. Bernstein, S. Jeffery, T. Lange, and A. Meurer. "Quantum Algorithms for the Subset-Sum Problem". In: *Proceedings of the 5th International Workshop on Post-Quantum Cryptography (PQCrypto)*. 2013, pp. 16–33 (cit. on pp. 5, 8, 9).

[BLT15] H. Buhrman, B. Loff, and L. Torenvliet. "Hardness of Approximation for Knapsack Problems". In: *Theory of Computing Systems* 56.2 (2015), pp. 372–393 (cit. on p. 2).

[Bri17] K. Bringmann. "A Near-Linear Pseudopolynomial Time Algorithm for Subset Sum". In: *Proceedings of the 28th Symposium on Discrete Algorithms (SODA)*. 2017, pp. 1073–1084 (cit. on p. 2).

[BST02] C. Bazgan, M. Santha, and Z. Tuza. "Efficient Approximation Algorithms for the Subset-Sums Equality Problem". In: *Journal of Computer and System Sciences* 64.2 (2002), pp. 160–170 (cit. on p. 8).

[CI21] J. Cardinal and J. Iacono. "Modular Subset Sum, Dynamic Strings, and Zero-Sum Sets". In: *Proceedings of the 4th Symposium on Simplicity in Algorithms (SOSA)*. 2021, pp. 45–56 (cit. on p. 2).

[CJRS22] X. Chen, Y. Jin, T. Randolph, and R. A. Servedio. "Average-Case Subset Balancing Problems". In: *Proceedings of the 33rd Symposium on Discrete Algorithms (SODA)*. 2022, pp. 743–778 (cit. on p. 8).

[EM20] A. Esser and A. May. "Low Weight Discrete Logarithm and Subset Sum in $2^{0.65n}$ with Polynomial Memory". In: *Proceedings of the 39th International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*. 2020, pp. 94–122 (cit. on p. 8).

[Gal68] R. G. Gallager. *Information Theory and Reliable Communication*. John Wiley and Sons, 1968 (cit. on p. 9).

[HJ10] N. Howgrave-Graham and A. Joux. "New Generic Algorithms for Hard Knapsacks". In: *Proceedings of the 29th International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*. 2010, pp. 235–256 (cit. on pp. 4, 8).

[HM18] A. Helm and A. May. "Subset Sum Quantumly in 1.17^n". In: *Proceedings of the 13th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC)*. 2018, 5:1–5:15 (cit. on p. 8).

[HS74] E. Horowitz and S. Sahni. "Computing Partitions with Applications to the Knapsack Problem". In: *Journal of the ACM* 21.2 (1974), pp. 277–292 (cit. on pp. 1, 2, 5).

[HW75] G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers*. Fourth. Oxford, 1975 (cit. on p. 9).

[Jef14] S. Jeffery. “Frameworks for Quantum Algorithms”. PhD thesis. University of Waterloo, 2014 (cit. on p. 12).

[JLL16] K. Jansen, F. Land, and K. Land. “Bounding the Running Time of Algorithms for Scheduling and Packing Problems”. In: *SIAM Journal on Discrete Mathematics* 30.1 (2016), pp. 343–366 (cit. on p. 2).

[Kar72] R. M. Karp. “Reducibility Among Combinatorial Problems”. In: *Proceedings of the Symposium on the Complexity of Computer Computations*. 1972, pp. 85–103 (cit. on p. 2).

[KPP04] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, 2004 (cit. on p. 3).

[MNPW19] M. Mucha, J. Nederlof, J. Pawlewicz, and K. Wegrzycki. “Equal-Subset-Sum Faster Than the Meet-in-the-Middle”. In: *Proceedings of the 27th European Symposium on Algorithms (ESA)*. 2019, 73:1–73:16 (cit. on pp. 1, 3–5, 7, 8, 17, 25, 27).

[MNRS11] F. Magniez, A. Nayak, J. Roland, and M. Santha. “Search via Quantum Walk”. In: *SIAM Journal on Computing* 40.1 (2011), pp. 142–164 (cit. on p. 12).

[MP91] N. Megiddo and C. H. Papadimitriou. “On Total Functions, Existence Theorems and Computational Complexity”. In: *Theoretical Computer Science* 81.2 (1991), pp. 317–324 (cit. on p. 3).

[Pap90] C. H. Papadimitriou. “On Graph-Theoretic Lemmata and Complexity Classes (Extended Abstract)”. In: *Proceedings of the 31st Symposium on Foundations of Computer Science (FOCS)*. 1990, pp. 794–801 (cit. on p. 3).

[SZZ18] K. Sotiraki, M. Zampetakis, and G. Zirdelis. “PPP-Completeness with Connections to Cryptography”. In: *Proceedings of the 59th Symposium on Foundations of Computer Science (FOCS)*. 2018, pp. 148–158 (cit. on p. 8).

[Tan09] S. Tani. “Claw Finding Algorithms Using Quantum Walk”. In: *Theoretical Computer Science* 410.50 (2009), pp. 5285–5297 (cit. on pp. 6, 11).

[Woe08] G. J. Woeginger. “Open Problems around Exact Algorithms”. In: *Discrete Applied Mathematics* 156.3 (2008), pp. 397–405 (cit. on p. 2).

[WY92] G. J. Woeginger and Z. Yu. “On the Equal-Subset-Sum Problem”. In: *Information Processing Letters* 42.6 (1992), pp. 299–302 (cit. on p. 2).

A Classical algorithms for Shifted-Sums

Here we adapt the classical $O(2^{0.773n})$ algorithm for Equal-Sums of Mucha et al. [MNPW19] to apply to the more general Shifted-Sums problem (Problem 6), with the same worst-case running time.

Theorem A.1 (Shifted-Sums, classical meet-in-the-middle)**.** *Given an instance of* Shifted-Sums *with maximum solution ratio* $\ell \in (0,1)$, *Algorithm* 6 *finds a solution with at least inverse polynomial probability in time* $\widetilde{O}(2^{n(h(\ell)+\ell)/2})$.

Proof. By the the same proof technique as Lemma 5.6, it follows that with at least inverse polynomial probability the random partition $X_1 \cup X_2$ satisfies $|(S_1 \cup S_2) \cap X_1| = |(S_1 \cup S_2) \cap X_2| = \ell n/2$. If this is the case, the algorithm will succeed.

The sets V_1, V_2 have cardinality $|V_1| = |V_2| = \binom{n/2}{\ell n/2} 2^{\ell n/2}$. Computing and sorting V_1, V_2 thus takes time $\widetilde{O}(\binom{n/2}{\ell n/2} 2^{\ell n/2})$. For each element $v_2 \in V_2$, binary search over V_1 takes times logarithmic in $|V_1|$. The result follows from Fact 2.1. $\square$

Algorithm 6: Classical meet-in-the-middle technique for SHIFTED-SUMS

Input: Instance (a, s) of SHIFTED-SUMS with maximum solution ratio ℓ.

Output: Two subsets $S_1, S_2 \subseteq [n]$.

1 Randomly split $[n]$ into disjoint subsets $X_1 \cup X_2$ such that $|X_1| = |X_2| = n/2$.

2 Classically compute and sort $V_1 = \{\Sigma(S_{11}) - \Sigma(S_{21}) : S_{11}, S_{21} \subseteq X_1$ and $S_{11} \cap S_{21} = \emptyset$ and $|S_{11}| + |S_{21}| = \ell n/2\}$.

3 Classically compute $V_2 = \{\Sigma(S_{12}) - \Sigma(S_{22}) : S_{12}, S_{22} \subseteq X_2$ and $S_{12} \cap S_{22} = \emptyset$ and $|S_{12}| + |S_{22}| = \ell n/2\}$

4 For each $v_2 \in V_2$, binary search for $v_1 \in V_1$ such that $v_1 + v_2 = s$. If such a v_2 is found, output $S_1 = S_{11} \cup S_{12}$ and $S_2 = S_{21} \cup S_{22}$, where $S_{11}, S_{21} \subseteq X_1$ are such that $v_1 = \Sigma(S_{11}) - \Sigma(S_{21})$ and S_{21}, S_{22} are such that $v_2 = \Sigma(S_{21}) - \Sigma(S_{22})$.

Algorithm 7: Classical representation technique for SHIFTED-SUMS

Input: Instance (a, s) of SHIFTED-SUMS with $\sum_{i=1}^n a_i < 2^{4n}$ and maximum solution ratio ℓ.

Output: Two subsets $S_1, S_2 \subseteq [n]$.

1 Set $b = 1 - \ell$ if $\ell > 1/2$ and $b = 1/2$ otherwise.

2 Choose a random prime $p \in [2^{bn} \,..\, 2^{bn+1}]$ and a random integer $k \in [0 \,..\, p-1]$.

3 Construct the table $t_p[i, j]$ for $i = 0, \ldots, n$ and $j = 0, \ldots, p-1$ (see Section 3).

4 Enumerate $T_{p,k}$ and $T_{p,(k-s) \bmod p}$, and sort $T_{p,(k-s) \bmod p}$. For each $S_1 \in T_{p,k}$, binary search for $S_2 \in T_{p,(k-s) \bmod p}$ such that $\Sigma(S_1) = \Sigma(S_2) + s$ and $S_1 \neq S_2$. If found, output the pair (S_1, S_2).

Theorem A.2 (SHIFTED-SUMS, classical representation)**.** *Given an instance of* SHIFTED-SUMS *with $\sum_{i=1}^n a_i < 2^{4n}$ and maximum solution ratio $\ell \in (0, 1)$,* Algorithm 7 *finds a solution with inverse polynomial probability in time $\widetilde{O}(2^{bn} + 2^{(1-b)n})$, where $b = 1 - \ell$ if $\ell > 1/2$ and $b = 1/2$ otherwise.*

Proof. The choice of b satisfies $b \leq 1 - \ell$. By Lemmas 3.10 and 5.4, with probability $\Omega(1/n)$ there is at least one solution pair contained in $T_{p,k} \times T_{p,(k-s) \bmod p}$. By Lemma 3.8 and Markov's inequality, the sizes of $T_{p,k}$ and $T_{p,(k-s) \bmod p}$ are at most $t_{p,k}, t_{p,(k-s) \bmod p} \leq n^2 2^{(1-b)n}$ with probability at least $1 - 1/n^2$. Thus, with probability $\Omega(1/n)$ we can assume that both of these events occur. If this is the case, then enumeration and sorting of $T_{p,k}$, $T_{p,(k-s) \bmod p}$ can be completed in time $\widetilde{O}(2^{(1-b)n})$ (Theorem 3.5) after constructing the table t_p in time $\widetilde{O}(2^{bn})$ (Lemma 3.2). $\square$

For each value of ℓ, choosing the better of Algorithms 6 and 7 gives the following result.

Theorem A.3 (SHIFTED-SUMS, classical)**.** *There is a classical algorithm that, given an instance of* SHIFTED-SUMS *with maximum solution ratio $\ell \in (0, 1)$, outputs a solution with at least inverse polynomial probability in time $\widetilde{O}(2^{\gamma(\ell)n})$ where*

$$\gamma(\ell) = \begin{cases} 1/2 & \text{if } \ell_1 \leq \ell < 1/2, \\ \ell & \text{if } 1/2 \leq \ell < \ell_2, \\ (h(\ell) + \ell)/2 & \text{otherwise} \end{cases}$$

and $\ell_1 \approx 0.227$ and $\ell_2 \approx 0.773$ are solutions to the equations $(h(\ell)+\ell)/2 = 1/2$ and $(h(\ell)+\ell)/2 = \ell$ respectively. In particular, the worst-case complexity is $O(2^{0.773n})$.

In comparison with the above result, the algorithm of [MNPW19] for Equal-Sums has running time $\widetilde{O}(2^{\gamma(\ell')n})$ where ℓ' is the minimum solution ratio (rather than maximum), and $\gamma(\ell') = \ell'$ for $1/2 \leq \ell' < \ell_2$ and $\gamma(\ell') = (h(\ell') + \ell')/2$ otherwise. We do not know if a similar algorithm exists for Shifted-Sums based on the minimum solution ratio.

B Quantum Equal-Sums in terms of minimum solution ratio

We first recall the concept of a *minimum solution*, introduced in [MNPW19].

Definition B.1 (Minimum solution)**.** Two disjoint subsets $S_1, S_2 \subseteq \{1, \ldots, n\}$ that form a solution to an instance of Equal-Sums are a *minimum solution* if their size $|S_1| + |S_2| = \ell'$ is the smallest among all such solutions. We call $\ell' \in (0, 1)$ the minimum solution ratio.

We prove that, for the special case of Equal-Sums (Problem 3), we can reformulate the results of Section 5 to make use of the minimum solution ratio ℓ' instead of the maximum one.

Theorem B.2 (Equal-Sums, quantum)**.** *There is a quantum algorithm that, given an instance of* Equal-Sums *with minimum solution ratio $\ell' \in (0, 1)$, outputs a solution with at least inverse polynomial probability in time $\widetilde{O}(2^{\gamma'(\ell')n})$ where*

$$\gamma'(\ell') = \begin{cases} \frac{1}{2} - \frac{1-\ell'}{4} h\left(\frac{\ell'}{2(1-\ell')}\right) & \text{if } \ell'_1 \leq \ell' < 1/2 \\ (1+\ell')/4 & \text{if } 1/2 \leq \ell' \leq 3/5, \\ \ell'/2 + 1/10 & \text{if } 3/5 < \ell' < \ell'_2, \\ (h(\ell') + \ell')/3 & \text{otherwise} \end{cases}$$

and $\ell'_1 \approx 0.273$ and $\ell'_2 \approx 0.809$ are solutions to the equations $(h(\ell') + \ell')/3 = 1/2 - (1 - \ell')h\left(\frac{\ell'}{2(1-\ell')}\right)/4$ and $(h(\ell') + \ell')/3 = \ell'/2 + 1/10$ respectively. In particular, the worst-case complexity is $O(2^{0.504n})$.

The proof follows closely that of the quantum algorithm for Shifted-Sums (Theorem 5.2), with the main difference coming from a bound on the size of the collision values set $V = \{v \in \mathbb{N} : \exists S_1 \neq S_2, v = \Sigma(S_1) = \Sigma(S_2)\}$ for Equal-Sums. For Shifted-Sums, Lemma 5.4 gives the bound $|V| \geq 2^{(1-\ell)n}$ in terms of the *maximum solution ratio* ℓ. For Equal-Sums we can obtain a similar statement in terms of the *minimum solution ratio*.

Lemma B.3. *If an instance of* Equal-Sums *has minimum solution ratio ℓ' then the collision values set $V = \{v \in \mathbb{N} : \exists S_1 \neq S_2, v = \Sigma(S_1) = \Sigma(S_2)\}$ satisfies*

$$|V| \geq \begin{cases} 2^{(1-\ell')n} & \text{if } \ell' > 1/2, \\ 2^{(1-\ell')h\left(\frac{\ell'}{2(1-\ell')}\right)n} & \text{otherwise.} \end{cases}$$

Proof. The case $\ell' > 1/2$ is dealt with in [MNPW19], therefore consider $\ell' \leq 1/2$. Let $S_1, S_2 \subseteq \{1, \ldots, n\}$ be a minimum solution of size $\ell' n$. Then for any $S \subseteq \overline{S_1 \cup S_2}$, with $|S| = \frac{\ell' n}{2} - 1$, the sets $S \cup S_1$ and $S \cup S_2$ form a solution, and for $S \neq S'$, the values $\Sigma(S_1 \cup S)$ and $\Sigma(S_1 \cup S')$ are distinct. Indeed, if this were not the case then $S \setminus S'$ and $S' \setminus S$ would form a disjoint solution of size less than $\ell' n$. Therefore $|V| \geq \binom{n(1-\ell')}{\frac{\ell' n}{2}-1}$, and the statement follows from Fact 2.1. ☐

We can now prove Theorem B.2.

Proof. For each minimum solution ratio $\ell' \in (0, 1)$, we use the better of Algorithms 4 and 5 with two modifications: (i) the minimum solution ratio ℓ' is used in place of the maximum solution ratio ℓ in the input of the algorithms, and (ii) we choose the value of b in step 1 of Algorithm 4 to be

$$b(\ell') = \begin{cases} \frac{1}{2} - \frac{1-\ell'}{4} h\big(\frac{\ell'}{2(1-\ell')}\big) & \text{if } \ell' \leq 1/2, \\ (1+\ell')/4 & \text{if } 1/2 < \ell' \leq 3/5, \\ 2/5 & \text{if } 3/5 < \ell'. \end{cases}$$

The analysis of Algorithm 5 is unaffected by the change to minimum solution ratio, as is the analysis of Algorithm 4 for $\ell' > 1/2$. For Algorithm 4 and $\ell' \leq 1/2$, repeating the analysis of Lemma 3.10 using $|V| \geq 2^{(1-\ell')h\big(\frac{\ell'}{2(1-\ell')}\big)n}$ gives $\Pr_{p,k}[v_{p,k} \geq 2^{(z-b)n-2}] = \Omega(1/n)$ where $z = (1-\ell')h\big(\frac{\ell'}{2(1-\ell')}\big)$. Recalling the proof of Theorem 5.5, the construction of the dynamic programming table takes time $\widetilde{O}(2^{bn})$, and a collision can be found in time $\widetilde{O}(t_{p,k}^{2/3}/v_{p,k}^{1/3}) = \widetilde{O}(2^{(2-z-b)/3})$. The running time for $\ell' < 1/2$ follows from balancing these two costs, i.e. by setting $b = (2-z-b)/3$. $\square$