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

Jonathan Allcock
Tencent Quantum Laboratory, Hong Kong, China

Yassine Hamoudi
Simons Institute for the Theory of Computing, University of California, Berkeley, CA, USA

Antoine Joux
CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

Felix Klingelhöfer
G-SCOP, Université Grenoble Alpes, France

Miklos Santha
Centre for Quantum Technologies and MajuLab, National University of Singapore, Singapore

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 $O(np)$, after which queries can be made in time $O(n)$ 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 on the best known $O(2^{0.773n})$ classical running time established by Mucha et al. [27]. We also study Pigeonhole Equal-Sums, a variant of Equal-Sums where the existence of a solution is guaranteed by the pigeonhole principle. For this problem we give faster classical and quantum algorithms with running time $\widetilde{O}(2^{n/2})$ and $\widetilde{O}(2^{2n/5})$, respectively.

2012 ACM Subject Classification Theory of computation → Quantum computation theory

Keywords and phrases Quantum algorithm, classical algorithm, dynamic programming, representation technique, subset-sum, equal-sum, shifted-sum

Digital Object Identifier 10.4230/LIPIcs.ESA.2022.6

Related Version *Full Version*: https://arxiv.org/abs/2111.07059 [3]

Funding This work has been supported by the European Union's H2020 Programme under grant agreement number ERC-669891, the ERA-NET Cofund in Quantum Technologies project QuantAlgo and the French ANR Blanc project RDAM. 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.

Acknowledgements JA thanks Shengyu Zhang for helpful discussions during the course of this work.

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 [23] 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 [19]. 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 $m^{o(1)}2^{o(n)}$ [13, 21]. 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 [12]. 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 $O(m^{1-\epsilon}2^{n\delta})$ [2]. On a quantum computer, meet-in-the-middle can be combined with Quantum Search to solve Subset-Sum in time $\widetilde{O}(2^{n/3})$.

1.1 Some Variants of Subset-Sum

Subset-Sum has several close relatives we will be concerned with in this paper.

▶ **Problem 2** (Equal-Sums [32])**.** 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 [31] 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 running time $\widetilde{O}(3^{n/3}) \leq O(2^{0.529n})$. The classical running time of Equal-Sums was reduced in a recent work by Mucha et al. [27] to $O(2^{0.773n})$, and it is an open problem whether this can be further improved.

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 [24]). In particular, we will study the case when every item can be used at most twice.

▶ **Problem 3** (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 4** (Shifted-Sums)**.** Given a multiset $\{a_1, \ldots, a_n\}$ of positive integers and a shift $0 \leq s < \sum_{i=1}^{n} a_i$, 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 the full version of the paper [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 a quantum algorithm for Shifted-Sums.

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 [26] consisting of NP-search problems with total relations. Problems in TFNP cannot be NP-hard unless NP equals co-NP. More precisely, the following problem belongs to the Polynomial Pigeonhole Principle complexity class PPP, defined by Papadimitriou [28], where the totality of the problem is syntactically guaranteed by the pigeonhole principle.

▶ **Problem 5** (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.

1.2 Our Contributions and Techniques

We give new classical and quantum algorithms for Subset-Sum and several closely related problems. Our main contribution is a quantum algorithm for Shifted-Sums (and for its special cases of Equal-Sums and 2-Subset-Sum) that improves on the currently best known $O(2^{0.773n})$ classical algorithm [27] and on the folklore $O(2^{0.529n})$ quantum meet-in-the-middle algorithm. We also initiate the study of the Pigeonhole Equal-Sums problem in the classical and quantum settings, where we obtain better complexities than what is known for the general Equal-Sums problem.

▶ **Theorem** (Theorem 18 (Restated))**.** *There is a quantum algorithm for* Shifted-Sums *that runs in time* $O(2^{0.504n})$.

▶ **Theorem** (Theorems 25, 26 (Restated))**.** *There are classical and quantum algorithms for* Pigeonhole Equal-Sums *that run in time* $\widetilde{O}(2^{n/2})$ *and* $\widetilde{O}(2^{2n/5})$*, respectively.*

In the full version of the paper [3], we give 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 [19]. We also describe partial results and potential directions for future work on modular versions of the problems studied in this paper.

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 [20], here we follow the path of Mucha et al. [27] and use it in a

worst case analysis. Among our new algorithms, the quantization of this technique for the Shifted-Sums problem is the most challenging and requires using several quantum tools. 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 $T_{p,k} = \left\{S \subseteq \{1,\ldots,n\} : \sum_{i\in S} a_i \equiv k \pmod p\right\}$, and we search that bin and $T_{p,(k-s) \bmod p}$ for a colliding solution (i.e. $(S_1, S_2) \in T_{p,k} \times T_{p,(k-s) \bmod p}$ such that $\sum_{i\in S_1} a_i = s + \sum_{i\in S_2} a_i$). The choice of the bin size (which, 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 $X \subseteq \overline{S_1 \cup S_2}$, the pairs $S_1 \cup X, S_2 \cup X$ also form a solution, and all these solutions have different values (see Lemma 20). 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 by the quantum element distinctness algorithm of Ambainis [4]. 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 $1 \le I \le |T_{p,k}|$. In other words, given $1 \le I \le |T_{p,k}|$, 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 $t_p[i,j] = \left|\left\{S \subseteq \{1,\ldots,i\} : \sum_{s\in S} a_s \equiv j \pmod p\right\}\right|$. Computing the cardinality of the bins is cheaper than computing their contents, and can be done in time $O(np) = \widetilde{O}(2^{2n/5})$. 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 $O(n)$. More precisely, we define a strict total order $\prec$ over $\mathcal{P}([n])$ and prove:

▶ **Theorem** (Theorem 11 (Restated)). *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 $I \in \{1, \ldots, |T_{p,k}|\}$ and random access to the elements of the table t_p, the set S_I can be computed in time $O(n)$.*

This novel data structure will be used in our algorithms for Subset-Sum (see the full version [3]), Shifted-Sums and Pigeonhole Equal-Sums. We now describe the additional quantum tools we use for Shifted-Sums. The algorithm randomly chooses a bin size of about $2^{(1-b)n}$ where b is defined differently depending on whether ℓ is above or below $3/5$, as different tools are required in these two regions. When $\ell \leq 3/5$, with high probability a random bin contains 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. Our algorithm generalizes the element distinctness [4] and claw finding [30] algorithms to the case of multiple marked pairs. Using an appropriate variant of the birthday paradox (see Lemma 5) we prove:

▶ **Theorem** (Theorem 6 (Quantum Pair Finding - Restated)). *Consider two sets of $N \leq M$ 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 $N \leq M \leq KN^2$ and $\widetilde{O}\big((M/K)^{1/2}\big)$ if $M \geq KN^2$.*

The best complexity when $\ell \leq 3/5$ 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 $b \leq 1-\ell$, 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 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 bins $T_{p,k}$ simultaneously satisfy two conditions: beside containing a solution, they should also have sizes 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 happen 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 [5], and achieve the same running time as given by the above argument. We believe that this 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 which 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 of p, if a bin $T_{p,k}$ has size larger than $2^n/p$ then it must contain a solution (see Lemma 24). 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 5).

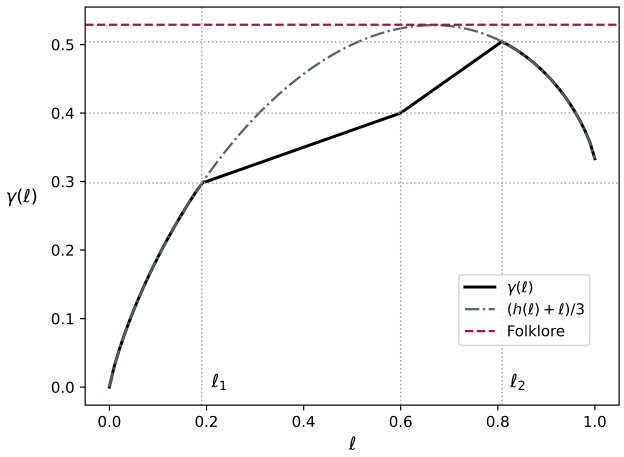

Figure 1 Running time exponent $\gamma(\ell)$ of the quantum Shifted-Sums algorithm, as a function of the maximum solution ratio ℓ (see Theorem 18). The maximum value of $\gamma(\ell)$ is ≈ 0.504 which occurs at $\ell = \ell_2 \approx 0.809$. For reference, the curve $(h(\ell) + \ell)/3$ corresponding to Theorem 23 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.

1.3 Related Works

The closest work to our contribution is the paper of Mucha et al. [27] 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 which is based around the concept of the size of a maximum solution, the classical algorithm is analyzed as 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 [27] 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 [20] 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, [20] can solve Subset-Sum instances in time $O(2^{0.337n})$. Since then, several variants of this classical method have been proposed [8, 15, 10, 14], while others have investigated quantum algorithms based on the representation technique. Bernstein et al. [9] improved on [20] 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 [18] and [10]. However, we emphasize that the algorithms in all these papers work for random inputs generated from some distributions. The paper [27] 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. [1]) but beyond that few other quantum improvements are known. One notable exception is the work of Ambainis et al. [6] 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 compete problem for the class was only identified relatively recently [29]. 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 [7] 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 Structure of the Paper

In Section 2.1, we define the notations and provide some basic facts. The quantum computational model and the algorithmic primitives used in the paper, such as our Quantum Pair Finding algorithm, are given in Section 2.2. In Section 3.1, we introduce the dynamic programming data structure and show how it can be used to implement fast subset-sum queries. The properties of this data structure needed for our work are proved in Section 3.2. The main application to the Shifted-Sums problem is described in Section 4. Finally, we study the pigeonhole variant of Equal-Sums in Section 5.

2 Preliminaries

2.1 Notations and Basic Facts

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 $\{m, m+1, \ldots n\}$, and by $[n]$ the set $[1\,..\,n]$. For sets $S, S' \subseteq [n]$ we denote by $\bar{S}$ the set $[n] \setminus S$, and by $S \Delta S'$ the symmetric difference of S and S'. Given a multiset $A = \{a_1, \ldots, a_n\}$ and 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 $\{0, \ldots, p-1\}$ which is congruent to a modulo p. The binary entropy function will be denoted by $h(x) = -x\log_2(x) - (1-x)\log_2(1-x)$.

▶ **Fact 1** ([16], 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.** *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 [17, 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. ◀

2.2 Quantum Algorithms

Quantum Computational Model. Similar to previous works on quantum element distinctness [4], quantum dynamic programming [6] and quantum subset sum algorithms [9], 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 2 for Shifted-Sums since multiple bins $T_{p,k}$ must be computed and stored in superposition.

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

▶ **Fact 3** (Quantum Search, Theorem 3 in [11]). *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 4** (Variable-Time Amplitude Amplification, Theorem 2 in [5]). *Let $\mathcal{A}$ be a quantum algorithm which 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 5** (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 [4] and claw finding [30] algorithms to the case of $K > 1$.

▶ **Theorem 6** (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 $\widetilde{O}\big((NM/K)^{1/3} \cdot \tau\big)$ if $N \leq M \leq KN^2$ and $\widetilde{O}\big((M/K)^{1/2} \cdot \tau\big)$ if $M \geq KN^2$.

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 5. Using the MNRS framework [25], 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 [4, Section 6.2] or [22, 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 3. ◀

3 Dynamic Programming Data Structure

3.1 Construction of the Data Structure

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

▶ **Definition 7.** *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 8.** *Let n, p be non-negative integers. In time $O(np)$, the $(n+1) \times p$ table $t_p[i, j] = |\{S \subseteq \{1, \ldots, i\} : \Sigma(S) \equiv j \pmod p\}|$ can be constructed by dynamic programming, where $i \in [0\,..\,n]$ and $j \in [0\,..\,p-1]$.*

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 $O(p)$ and the computation of all rows can be completed in time $O(np)$. ◀

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 9.** *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 10.** *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 $O(n)$.

▶ **Theorem 11.** *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 $O(n)$.*

Proof. Algorithm 1 gives a process which 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.

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

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 = ∅.
for i = n, ..., 1 do
    if I ≤ t_p[i − 1, j] then
        Do nothing.
    else
        Update Z = Z ∪ {i}, I = I − t_p[i − 1, j] and j = j − a_i mod p.
Return Z.
```

Here, we provide a high level explanation of why the algorithm works, and defer a formal proof to the full version of the paper [3]. 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 $n-1$, $\{S \subseteq [n-1] : \Sigma(S) \equiv k - a_n \pmod p\}$ which 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. ◀

▶ **Corollary 12** (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 $c \le |T_{p,k}|$, it is possible to find c elements of $T_{p,k}$ in time $\widetilde{O}(p+c)$.*

Proof. By Lemma 8 the table $t_p[i,j]$ can be constructed in time $O(np)$. Thereafter, by Theorem 11 each set S_I for $I \in [1\,..\,t_{p,k}]$ can be computed in time $O(n)$. ◀

An alternative method for enumerating solutions was previously known:

▶ **Fact 13** (Enumerating solutions to Subset-Sum [8]). *Let $A = \{a_1, \ldots, a_n\}$ where $a_i = 2^{O(n)}$ for all i. Let $p = 2^{O(n)}$, $0 \le k \le p-1$, and $T_{p,k} = \{S \subseteq [n] : \Sigma_A(S) \equiv k \pmod p\}$. Then, for any $c \le |T_{p,k}|$, it is possible to find c elements of $T_{p,k}$ in time $\widetilde{O}(2^{n/2} + c)$.*

In comparison with Fact 13, 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 7) 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.

▶ **Lemma 14.** *The expected bin size can be upper bounded as $\mathbb{E}_{p,k}[t_{p,k}] \le 2^{(1-b)n}$.*

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

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 analyze the complexity of the Variable-Time Amplitude Amplification algorithm.

▶ **Lemma 15.** *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 $\mathbb{E}_{p,k}[t_{p,k}t_{p,(k-s) \bmod p}] = \mathbb{E}_{p,k}\left[\sum_{S_1,S_2} \mathbb{1}_{\Sigma(S_1) \equiv_p \Sigma(S_2)+s \equiv_p k}\right]$. 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. 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 16.** *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 15, 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. 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 Shifted-Sums

Our approach for solving Shifted-Sums relies on two different quantum algorithms. In Section 4.1, we present the first algorithm (based on the representation technique), which is more involved. The description of the second algorithm (based on meet-in-the-middle) is given in Section 4.2. The running time of both algorithms, expressed in Theorems 21 and 23, are functions of the size of a *maximum solution* of the input.

▶ **Definition 17** (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 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 the following performance:

▶ **Theorem 18** (Shifted-Sums, quantum). *There is a quantum algorithm which, 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*

$$\gamma(\ell) = \begin{cases} (1+\ell)/4 & \text{if } \ell_1 \le \ell \le 3/5, \quad \text{(Theorem 21)} \\ \ell/2 + 1/10 & \text{if } 3/5 < \ell < \ell_2, \quad \text{(Theorem 21)} \\ (h(\ell)+\ell)/3 & \text{otherwise} \quad \text{(Theorem 23)} \end{cases}$$

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.

4.1 Representation Technique Algorithm

Our representation technique based algorithm is given in Algorithm 2, 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 4) on top of Quantum Pair Finding (Theorem 6).

Algorithm 2 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 3) 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 4) on steps **4** - **6**, otherwise run them once:

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

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 6) to find if there exists 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 4 to the algorithm $\mathcal{A}$ defined by the following process (i) create a uniform superposition over all primes $p \in [2^{bn} \,..\, 2^{bn+1}]$ and, for each p, all $k \in [0 \,..\, p-1]$. (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 Algorithm 2 relies on the random bins statistics presented in Section 3.2. We first define the *collision values set* which contains the values of all the possible solution pairs.

▶ **Definition 19** (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 16, we can lower bound the number of values in V that get hashed to a random bin $T_{p,k}$.

▶ **Lemma 20.** *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 now prove the correctness of Algorithm 2.

▶ **Theorem 21** (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 2 *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 2 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 given in Theorem 21.

Analysis when $\ell \leq 3/5$. In this case the algorithm executes steps **4** - **6** only once. From Lemma 8, the table t_p can be constructed in time $O(n2^{bn})$, after which each query to the elements of $T_{p,k}$ can be performed in time $O(n)$ (Theorem 11). By Lemma 16, 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 14 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 6 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 20 and 16, 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 6 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 $\mathbb{E}[\tau^2] = \widetilde{O}(\mathbb{E}_{p,k}[(p + (t_{p,k} t_{p,(k-s) \bmod p})^{1/3} + \sqrt{\max(t_{p,k}, t_{p,(k-s) \bmod p})})^2]) \leq \widetilde{O}(\mathbb{E}_{p,k}[p^2] + \mathbb{E}_{p,k}[t_{p,k}^{2/3} t_{p,(k-s) \bmod p}^{2/3}] + \mathbb{E}_{p,k}[t_{p,k}]) \leq \widetilde{O}(\mathbb{E}_{p,k}\big[p^2\big] + \mathbb{E}_{p,k}[t_{p,k} t_{p,(k-s) \bmod p}]^{2/3} + \mathbb{E}_{p,k}[t_{p,k}]) \leq \widetilde{O}(2^{2bn}) + \widetilde{O}(2^{4(1-b)n/3}) + 2^{(1-b)n}$, where the second inequality uses that the moment function is non-decreasing and the last inequality uses Lemmas 14 and 15. Since $b = 2/5$ we obtain that $\mathbb{E}[\tau^2] \leq \widetilde{O}(2^{2bn})$. Finally, by Fact 4, the overall time of steps **3** - **6** is $\widetilde{O}(\sqrt{\mathbb{E}[\tau^2]/\rho}) = \widetilde{O}(2^{bn}/2^{(1-\ell-b)n/2}) = \widetilde{O}(2^{(\frac{\ell}{2}+\frac{1}{10})n})$. ◀

4.2 Quantum Meet-in-the-Middle Algorithm

We describe the second quantum algorithm for solving Shifted-Sums based on the meet-in-the-middle technique combined with Quantum Search. 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 22.** *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 1 gives that this quantity is at least $\Omega\big(n^{-1/2}\big)$. ◀

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

Algorithm 3 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 $V_1 = \{\Sigma(S_{11}) - \Sigma(S_{21}) : S_{11}, S_{21} \subseteq X_1$ and $S_{11} \cap S_{21} = \emptyset$ and $|S_{11}| + |S_{21}| = \ell n/3\}$.
3. Apply Quantum Search (Fact 3) over the set $V_2 = \{\Sigma(S_{12}) - \Sigma(S_{22}) : S_{12}, S_{22} \subseteq X_2$ and $S_{12} \cap S_{22} = \emptyset$ and $|S_{12}| + |S_{22}| = 2\ell n/3\}$, 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 23** (Shifted-Sums, meet-in-the-middle). *Given an instance of* Shifted-Sums *with maximum solution ratio $\ell \in (0, 1)$,* Algorithm 3 *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 takes 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 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 22, 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. ◀

5 Pigeonhole Equal-Sums

We give classical and quantum algorithms for the Pigeonhole Equal-Sums problem, 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 which 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 24.** *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 exists two distinct subsets S_1, S_2 with $\Sigma(S_1) \equiv \Sigma(S_2) \equiv k \pmod p$.*

Proof. Denote by $\overline{0}, \overline{1}, \dots \overline{p-1}$ 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 8, the table $t_p[i,j] = |\{S \subseteq \{1,\dots,i\} : \Sigma(S) \equiv j \pmod p\}|$ can be constructed in time $\widetilde{O}(p)$. From the table, we can read off a value k that satisfies the above condition. ◀

▶ **Theorem 25** (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 24, in time $\widetilde{O}(2^{n/2})$ we can find k such that there exists $S_1 \neq S_2$ 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 12, we can enumerate in time $\widetilde{O}(2^{n/2})$ a sufficient number of subsets in that bin to locate a collision. ◀

▶ **Theorem 26** (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 24, in time $\widetilde{O}(2^{2n/5})$ we can identify k such that there exists $S_1 \neq S_2$ satisfying $\Sigma(S_1) \equiv \Sigma(S_2) \equiv k \pmod{2^{2n/5}}$. By Theorem 11, each query to $T_{p,k} = \{S \subseteq [n] : \Sigma(S) \equiv k \pmod p\}$ can be made in time $O(n)$. We use Ambainis' element distinctness algorithm [4] 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 $\widetilde{O}(2^{2n/5})$. ◀

In the full version [3], we give an $\widetilde{O}(2^{n/2})$ classical deterministic algorithm for the following modular version of Pigeonhole Equal-Sums. We also ask the open question of finding a faster quantum algorithm.

▶ **Problem 6** (Pigeonhole Modular Equal-Sums)**.** Given a set $\{a_1, \dots, a_n\}$ 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 $\sum_{i \in S_1} a_i \equiv \sum_{i \in S_2} a_i \pmod q$.

References

1. A. Abboud. Fine-grained reductions and quantum speedups for dynamic programming. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 8:1–8:13, 2019.
2. 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)*, pages 41–57, 2019.
3. J. Allcock, Y. Hamoudi, A. Joux, F. Klingelhöfer, and M. Santha. Classical and quantum algorithms for variants of subset-sum via dynamic programming, 2021. `arXiv:2111.07059 [quant-ph]`.
4. A. Ambainis. Quantum walk algorithm for element distinctness. *SIAM Journal on Computing*, 37(1):210–239, 2007.
5. A. Ambainis. Variable time amplitude amplification and quantum algorithms for linear algebra problems. In *Proceedings of the 29th International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 636–647, 2012.
6. 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)*, pages 1783–1793, 2019.
7. C. Bazgan, M. Santha, and Z. Tuza. Efficient approximation algorithms for the Subset-Sums Equality problem. *Journal of Computer and System Sciences*, 64(2):160–170, 2002.
8. 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)*, pages 364–385, 2011.
9. 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)*, pages 16–33, 2013.
10. 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)*, pages 633–666, 2020.
11. M. Boyer, G. Brassard, P. Høyer, and A. Tapp. Tight bounds on quantum searching. *Fortschritte der Physik*, 46(4-5):493–505, 1998.
12. K. Bringmann. A near-linear pseudopolynomial time algorithm for subset sum. In *Proceedings of the 28th Symposium on Discrete Algorithms (SODA)*, pages 1073–1084, 2017.
13. H. Buhrman, B. Loff, and L. Torenvliet. Hardness of approximation for knapsack problems. *Theory of Computing Systems*, 56(2):372–393, 2015.
14. 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)*, pages 743–778, 2022.
15. 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)*, pages 94–122, 2020.
16. R. G. Gallager. *Information Theory and Reliable Communication*. John Wiley and Sons, 1968.
17. G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers*. Oxford, fourth edition, 1975.
18. 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)*, pages 5:1–5:15, 2018.
19. E. Horowitz and S. Sahni. Computing partitions with applications to the knapsack problem. *Journal of the ACM*, 21(2):277–292, 1974.
20. 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)*, pages 235–256, 2010.

21 K. Jansen, F. Land, and K. Land. Bounding the running time of algorithms for scheduling and packing problems. *SIAM Journal on Discrete Mathematics*, 30(1):343–366, 2016.

22 S. Jeffery. *Frameworks for Quantum Algorithms*. PhD thesis, University of Waterloo, 2014.

23 R. M. Karp. Reducibility among combinatorial problems. In *Proceedings of the Symposium on the Complexity of Computer Computations*, pages 85–103, 1972.

24 H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, 2004.

25 F. Magniez, A. Nayak, J. Roland, and M. Santha. Search via quantum walk. *SIAM Journal on Computing*, 40(1):142–164, 2011.

26 N. Megiddo and C. H. Papadimitriou. On total functions, existence theorems and computational complexity. *Theoretical Computer Science*, 81(2):317–324, 1991.

27 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)*, pages 73:1–73:16, 2019.

28 C. H. Papadimitriou. On graph-theoretic lemmata and complexity classes (extended abstract). In *Proceedings of the 31st Symposium on Foundations of Computer Science (FOCS)*, pages 794–801, 1990.

29 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)*, pages 148–158, 2018.

30 S. Tani. Claw finding algorithms using quantum walk. *Theoretical Computer Science*, 410(50):5285–5297, 2009.

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

32 G. J. Woeginger and Z. Yu. On the equal-subset-sum problem. *Information Processing Letters*, 42(6):299–302, 1992.