information

IMPACT FACTOR 2.9

CITESCORE 6.5

Article

# From Subset-Sum to Decoding: Improved Classical and Quantum Algorithms via Ternary Representation Technique

Yang Li

MDPI

https://doi.org/10.3390/info16100887

information

MDPI

*Article*

# From Subset-Sum to Decoding: Improved Classical and Quantum Algorithms via Ternary Representation Technique

**Yang Li**

Key Laboratory of System Software (Chinese Academy of Sciences) and State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China; liyang@ios.ac.cn

**Abstract**

The subset-sum problem, a foundational NP-hard problem in theoretical computer science, serves as a critical building block for cryptographic constructions. This work introduces novel classical and quantum heuristic algorithms for the random subset-sum problem at density $d = 1$, where exactly one solution is expected. Classically, we propose the first algorithm based on a ternary tree representation structure, inspired by recent advances in lattice-based cryptanalysis. Through numerical optimization, our method achieves a time complexity of $\widetilde{\mathcal{O}}(2^{0.2400n})$ and space complexity of $\widetilde{\mathcal{O}}(2^{0.2221n})$, improving upon the previous best classical heuristic result of $\widetilde{\mathcal{O}}(2^{0.2830n})$. In the quantum setting, we develop a corresponding algorithm by integrating the classical ternary representation technique with a quantum walk search framework. The optimized quantum algorithm attains a time and space complexity of $\widetilde{\mathcal{O}}(2^{0.1843n})$, surpassing the prior state-of-the-art quantum heuristic of $\widetilde{\mathcal{O}}(2^{0.2182n})$. Furthermore, we apply our algorithms to information set decoding in code-based cryptography. For half-distance decoding, our classical algorithm improves the time complexity to $\widetilde{\mathcal{O}}(2^{0.0453n})$, surpassing the previous best of $\widetilde{\mathcal{O}}(2^{0.0465n})$. For full-distance decoding, we achieve a quantum complexity of $\widetilde{\mathcal{O}}(2^{0.058326n})$, advancing beyond the prior best quantum result of $\widetilde{\mathcal{O}}(2^{0.058696n})$. These findings demonstrate the broad applicability and efficiency of our ternary representation technique across both classical and quantum computational models.

**Keywords:** subset-sum; information set decoding; representation technique; quantum algorithm; quantum walk



Academic Editor: Pingping Chen

Received: 11 September 2025
Revised: 6 October 2025
Accepted: 10 October 2025
Published: 12 October 2025

**Citation:** Li, Y. From Subset-Sum to Decoding: Improved Classical and Quantum Algorithms via Ternary Representation Technique. *Information* **2025**, *16*, 887. https://doi.org/10.3390/info16100887



## 1. Introduction

The subset-sum problem, also referred to as the knapsack problem, stands as one of the most fundamental and well-known NP-hard problems in theoretical computer science [1]. Owing to its computational intractability, it has been widely employed in the construction of cryptographic systems [2–5]. Moreover, this problem underpins several cryptographic schemes targeting post-quantum security (see, e.g., ref. [6]) and acts as a key component in certain quantum hidden shift algorithms [7]. These, in turn, are applied in quantum cryptanalysis of isogeny-based protocols [8] and symmetric cryptographic constructions [9].

An instance of the problem is specified by $n$ integers $(a_1, a_2, \ldots, a_n) \in (\mathbb{Z}_{2^n})^n$ and a target integer $s \in \mathbb{Z}_{2^n}$. There are two common formulations: the decision version, which is to verify the existence of a subset of $\{a_1, a_2, \ldots, a_n\}$ summing to $s$, and the computational version, which requires finding such a subset. The decision subset-sum problem is NP-complete. Moreover, given oracle access to the decision problem, a solution to the

computational version can be recovered with $n$ queries. When the instance is uniformly random, the problem is termed the random subset-sum problem (RSSP).

The *density* of an instance is defined as $d = n/\log_2(\max_i a_i)$. For a random instance $\mathbf{a}$, the density is closely related to the expected number of solutions. In the high-density regime where $d = \Omega(1/\log n)$, dynamic programming offers an efficient solution [10]. In the low-density setting, Brickell [11] and Lagarias and Odlyzko [12] demonstrated that given oracle access to the shortest vector problem in lattices [13], RSSP can be solved efficiently for densities $d < 0.64$. Coster et al. [14] and Joux and Stern [15] further raised this bound to $d < 0.94$. For knapsack instances with density approaching 1, the best known algorithms exhibit exponential-time complexity, rendering them computationally intractable; this inherent difficulty is the reason for their designation as "hard" knapsacks.

This work addresses the random subset-sum problem at density $d = 1$, where exactly one solution is expected to exist. Horowitz and Sahni's Meet-in-the-Middle approach [16] reduces the time and space complexity from $\mathcal{O}(2^n)$ to $\mathcal{O}(2^{n/2})$, avoiding the need for an exhaustive search over all $2^n$ subsets. The idea of the method is to find a collision between two lists of partial sums, each comprising $2^{n/2}$ elements. Schroeppel and Shamir [17] refined this approach with a strategy based on merging four lists, which lowered the space complexity to $\mathcal{O}(2^{n/4})$.

A significant heuristic algorithm was proposed by Howgrave-Graham and Joux (HGJ) [18] at EUROCRYPT 2010, solving the RSSP with a time complexity of $\widetilde{\mathcal{O}}(2^{0.337n})$ and a space complexity of $\widetilde{\mathcal{O}}(2^{0.311n})$, thus breaking the long-standing $2^{n/2}$ barrier. Their key innovation was the use of the *representation technique*, in which the solution is expressed ambiguously as a sum of vectors in $\{0,1\}^n$. This expands the size of the search space and enables the merging of more lists under arbitrary constraints, leading to improved time efficiency. The time and space complexity is derived via numerical optimization, under the standard heuristic assumption that the individual elements obtained in the merging steps are well-distributed. Becker, Coron, and Joux (BCJ) [19] subsequently reduced the time and space complexity to $\widetilde{\mathcal{O}}(2^{0.291n})$ by introducing representations using vectors from $\{-1,0,1\}^n$. Later, Bonnetain, Bricout, Schrottenloher, and Shen (BBSS) [20] further reduced the time and space complexity to $\widetilde{\mathcal{O}}(2^{0.283n})$ by relaxing BCJ constraints and permitting representations in $\{-1,0,1,2\}^n$. Not only has the representation technique been employed as a tool for the subset-sum problem but it has also been successfully applied to the syndrome decoding problem [21–24] and the learn-with-errors (LWE) problem [25–29].

Quantum algorithms for the subset-sum problem have also attracted significant attention. The first quantum speedup was achieved by Bernstein, Jeffery, Lange, and Meurer (BJLM) [30], who proposed an algorithm leveraging the HGJ technique, achieving a time and space complexity of $\widetilde{\mathcal{O}}(2^{0.241n})$. Helm and May (HM) [31] subsequently developed a quantum variant of the BCJ algorithm that obtains a time and space complexity of $\widetilde{\mathcal{O}}(2^{0.226n})$. Later, Bonnetain et al. [20] introduced a quantum algorithm based on their classical algorithm, attaining the time and space complexity of $\widetilde{\mathcal{O}}(2^{0.218n})$ under the same classical heuristic assumptions.

### *1.1. Our Contribution*

In this work, we introduce a new heuristic algorithm for the subset-sum problem based on a ternary representation technique. The algorithm is further accelerated within a quantum walk search framework, leading to a novel quantum approach.

Representation-based algorithms for the subset-sum problem typically enhance efficiency by increasing the number of representations. These methods inherently exhibit a search tree structure. By recursively constructing parent lists from child lists until reaching a root list that is expected to contain a solution, such algorithms effectively navigate the

problem space. To date, all known classical heuristic algorithms employ a binary tree structure. Notably, Lee et al. [29] demonstrated in the context of ternary LWE cryptanalysis that ternary trees significantly increase the number of representations compared to binary trees. Inspired by their idea, we propose the first classical heuristic algorithm for the random subset-sum problem (RSSP) based on a ternary tree structure. Through numerical optimization, we show that our algorithm achieves a time complexity of $\widetilde{\mathcal{O}}(2^{0.2400n})$ and a space complexity of $\widetilde{\mathcal{O}}(2^{0.2221n})$, which, to the best of our knowledge, represents the current state of the art among classical heuristic algorithms.

In the quantum setting, we develop a novel quantum algorithm by integrating the proposed classical heuristic with a quantum walk over the Johnson graph, under the MNRS (Magniez–Nayak–Roland–Santha) framework [32]. This quantum algorithm retains the same heuristic foundation as its classical counterpart. After numerical optimization, we demonstrate that it achieves a time and space complexity of $\widetilde{\mathcal{O}}(2^{0.1843n})$, establishing the best-known quantum heuristic performance for RSSP.

A summary of these complexity results is provided in Table 1.

**Table 1.** Comparison of complexity exponent of subset-sum algorithms: classical and quantum approaches.

| | Algorithm | Time Complexity | Space Complexity |
|---|---|---|---|
| Classical | HGJ | 0.337 | 0.311 |
| | BCJ | 0.291 | 0.291 |
| | BBSS | 0.2830 | 0.2830 |
| | Ours | **0.2400** | **0.2221** |
| Quantum | BJLM | 0.241 | 0.241 |
| | HM | 0.226 | 0.226 |
| | BBSS | 0.2182 | 0.2182 |
| | Ours | **0.1843** | **0.1843** |

*Note:* Bold values highlight the best (lowest) complexity exponents in each category.

Furthermore, we apply both our classical and quantum algorithms to the decoding problem. Code-based cryptography, which relies on the hardness of decoding random linear codes, is a major candidate for post-quantum cryptography (PQC). Three of the four encryption schemes chosen in the fourth round of the NIST PQC standardization project [33] are code-based schemes, namely, Classic McEliece [34], BIKE [35], and HQC [36]. Among these, only HQC was ultimately selected for standardization. This prominence underscores the importance of rigorous security analysis for such systems.

Our decoding algorithms operate within the framework of *information set decoding (ISD)*. The first classical ISD algorithm was introduced by Prange in 1962 [37]. Subsequent improvements were made by Stern [38] and Finiasz and Sendrier [39]. Later, further advancements were achieved through the use of representation techniques by May, Meurer, and Thomae (MMT) [21] as well as by Becker, Joux, May, and Meurer (BJMM) [22]. These were later enhanced by combining representation techniques with nearest neighbor searches, as proposed by May and Ozerov (MO) [40] and Both and May (BM) [41]. Currently, the Both–May algorithm [41] achieves the optimal time complexity among classical algorithms. Recent work has also explored time–memory trade-offs [42–44].

By applying our subset-sum algorithm to half-distance decoding, we improve upon the best-known ISD algorithm by Both–May, reducing the time complexity from $\widetilde{\mathcal{O}}(2^{0.0465n})$ to $\widetilde{\mathcal{O}}(2^{0.0453n})$. For full-distance decoding, our algorithm achieves a time complexity of $\widetilde{\mathcal{O}}(2^{0.0973n})$, improving upon the previous $\widetilde{\mathcal{O}}(2^{0.1020n})$ [22], which relied solely on representation techniques. Although this result does not surpass the state-of-the-art complexity of $\widetilde{\mathcal{O}}(2^{0.0885n})$ attained by BM [41]—which combines both representation and nearest neigh-

bor techniques—we anticipate that integrating our ternary representation approach with nearest neighbor search could lead to further improvements.

Quantum variants of ISD have been less extensively explored: Bernstein [45] analyzed a quantum version of Prange's algorithm, and Kachigar and Tillich (KT) [23] later proposed several quantum-walk-based ISD algorithms, which currently represent the fastest known quantum ISD algorithms. Kirshanova [24] provided an alternative perspective on the quantum-walk-based KT algorithms and developed a quantum version of the MO algorithm. In this work, we propose a novel quantum ISD algorithm that outperforms the state-of-the-art algorithm by Kachigar and Tillich. A summary of the complexity results for ISD is provided in Table 2.

**Table 2.** Comparison of time complexity exponent of ISD: classical and quantum approaches.

| | Algorithm | Full Distance | Half Distance |
|---|---|---|---|
| Classical | Prange | 0.1206 | 0.0576 |
| | MMT | 0.1116 | 0.0537 |
| | BJMM | 0.1019 | 0.0494 |
| | MO | 0.0953 | 0.0473 |
| | BM | **0.0885** | 0.0465 |
| | Ours | 0.0973 | **0.0453** |
| Quantum | Bernstein | 0.060350 | −− |
| | KT | 0.058696 | −− |
| | Kirshanova | 0.059450 | −− |
| | Ours | **0.058326** | −− |

*Note:* Bold values highlight the best (lowest) complexity exponents in each category.

### *1.2. Organization*

The remainder of this paper is structured as follows. Section 2 introduces essential preliminary concepts required for our subsequent analysis and illustrates representation techniques, in particular using the HGJ algorithm as an example. In Section 3, we first present our new classical heuristic subset-sum algorithm, detailing its procedure, explaining its correctness, and analyzing its complexity. Then, through rigorous mathematical derivation, we provide a provable probabilistic algorithm that does not rely on heuristics. In Section 4, we turn to quantum algorithms, explaining in detail how the newly proposed classical algorithm is combined with a quantum walk and subsequently analyzing the complexity of the resulting algorithm. In Section 5, we apply our new classical and quantum algorithms to the decoding problem, demonstrating how they can be incorporated into the information set decoding framework. Specifically, the quantum algorithm employs a nested structure combining Grover search and quantum walks. Finally, in Section 6, we conclude with a summary of our new results.

## 2. Preliminaries

This section introduces the foundational definitions and standard notations for the subset-sum problem, followed by an overview of representation techniques, illustrated through the HGJ algorithm.

### *2.1. Notations and Conventions*

**Definition 1** (Distributions of knapsacks)**.** *A knapsack (or subknapsack) is a vector* $\mathbf{e} \in D^n$ *where* $D = \{-1, 0, 1\}$ *is the digit set. We denote by* $D^n[\alpha, \beta]$ *the set of vectors* $\mathbf{e}$ *containing exactly* $\alpha n$ *entries equal to* $-1$, $(\alpha + \beta)n$ *entries equal to* $1$, *and* $(1 - 2\alpha - \beta)n$ *entries equal to* $0$.

**Definition 2** (Random subset-sum instance)**.** *Let* $\mathbf{a} \in (\mathbb{Z}_N)^n$ *be chosen uniformly at random, where* $N \simeq 2^n$. *For* $\beta \in [0, 1]$, *sample a vector* $\mathbf{e}$ *uniformly from* $D^n[0, \beta]$, *and define* $s \equiv$

$\langle \mathbf{a}, \mathbf{e} \rangle \bmod N$. *Then the pair* $(\mathbf{a}, s) \in (\mathbb{Z}_N)^{n+1}$ *is termed a random subset-sum instance. Any vector* $\mathbf{e}' \in \{0,1\}^n$ *satisfying* $\langle \mathbf{a}, \mathbf{e}' \rangle \equiv s \bmod N$ *is called a solution to the instance.*

It is well established that the parameter choice $\beta = 1/2$ represents the most computationally challenging scenario, as the vector $\mathbf{e}$ contains an equal proportion of zeros and ones, thereby maximizing combinatorial complexity. Consequently, this paper concentrates exclusively on this specific case.

In asymptotic analysis, we utilize the standard formula given below to estimate the cardinality of vector sets that contain a fixed quantity of certain coefficients.

**Lemma 1** (Multinomial approximation)**.** *Let* $Dig = \{d_1, \ldots, d_k\} \subset \mathbb{Z}_N$ *be a digit set of cardinality* $k$*. Consider vectors in* $\mathbb{Z}_N^n \cap Dig^n$ *where each digit* $d_i$ *appears exactly* $c_i n$ *times and the coefficients* $c_i$ *satisfy* $\sum_{i=1}^{k} c_i = 1$*. The number of such vectors is asymptotically given by the multinomial coefficient:*

$$\binom{n}{c_1 n, \cdots, c_k n} \approx 2^{H(c_1, \cdots, c_k) n},$$

*where the entropy function is defined as* $H(c_1, \cdots, c_k) := \sum_{i=1}^{k} c_i \log_2(\frac{1}{c_i})$.

We use the notation $\binom{n}{c_1 n, \cdots, c_{k-1} n, \cdot}$ as a shorthand for the multinomial coefficient $\binom{n}{c_1 n, \cdots, c_k n}$, with the dot $(\cdot)$ standing for the remaining term $n - \sum_{i=1}^{k-1} c_i n$. Similarly, we write $H(c_1, \ldots, c_{k-1}, \cdot)$ for the entropy $H(c_1, \ldots, c_k)$.

We adopt the soft-O notation, denoted by $\widetilde{\mathcal{O}}$, to suppress polylogarithmic factors in $n$. Any rounding is omitted for simplicity.

### 2.2. Representation Techniques

From a high-level perspective, classical algorithms based on representation techniques recursively generate subknapsack lists and merge them to ultimately obtain a list of solutions to the original knapsack problem.

We use the HGJ algorithm [18] as an example due to its seminal status and relatively simple structure. The tree structure of the HGJ algorithm is illustrated in Figure 1.

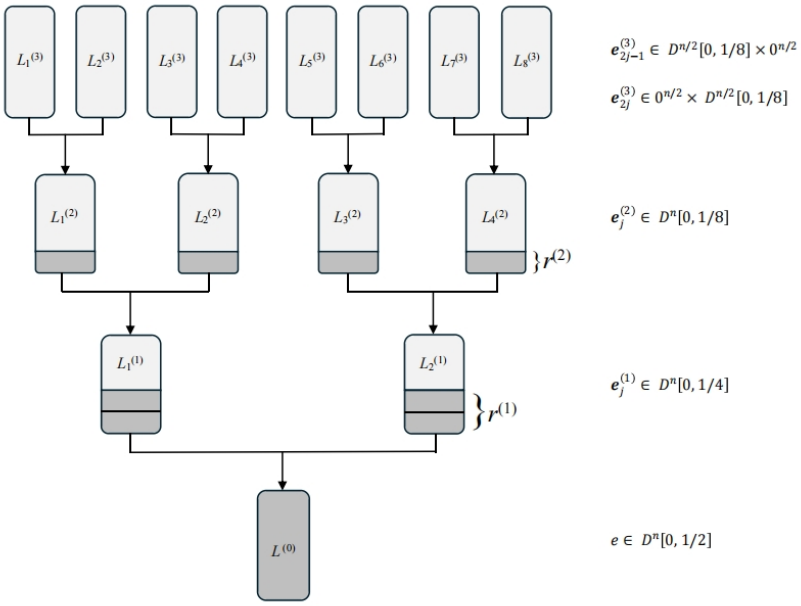


**Figure 1.** The tree structure of the HGJ algorithm. The darker regions indicate portions where the modulus constraint has been established.

Consider a subset-sum instance $(\mathbf{a}, s) \in (\mathbb{Z}_N)^{n+1}$ with a solution $\mathbf{e} \in D^n[0, 1/2]$, meaning $\langle \mathbf{a}, \mathbf{e} \rangle \equiv s \bmod 2^n$, and exactly $n/2$ components of $\mathbf{e}$ are 1, while the remaining $n/2$ are 0.

The core idea is to represent $\mathbf{e}$ ambiguously as a sum of vectors in $\{0,1\}^n$, i.e., $\mathbf{e} = \mathbf{e}_1^{(1)} + \mathbf{e}_2^{(1)}$ with $\mathbf{e}_1^{(1)}, \mathbf{e}_2^{(1)} \in D^n[0, 1/4]$. We call $(\mathbf{e}_1^{(1)}, \mathbf{e}_2^{(1)})$ a *representation* of $\mathbf{e}$. The ambiguity introduces $R^{(1)} = \binom{n/2}{n/4}$ representations of $\mathbf{e}$, since each of the $n/2$ 1-coordinates in $\mathbf{e}$ can be represented as $1 + 0$ or $0 + 1$. This paper adopts the notation where a parenthesized superscript corresponds to the depth of a given vector or parameter in the tree.

Since finding even a single representation from this exponential set is adequate to solve the subset-sum problem, the HGJ algorithm's core principle is to aim for the computation of approximately a $1/R^{(1)}$ fraction of all representations. This targeted approach is designed to yield, in expectation, exactly one valid representation.

This motivates the following strategy. Select a modulus $2^{r^{(1)}}$, where $r^{(1)} = \lfloor \log_2 R^{(1)} \rfloor$, and sample a target value at random $s_1^{(1)} \in \{0, 1, \ldots, 2^{r^{(1)}} - 1\}$. Define the lists

$$L_1^{(1)} = \left\{ \left( \mathbf{e}_1^{(1)}, \langle \mathbf{a}, \mathbf{e}_1^{(1)} \rangle \right) \in D^n[0, 1/4] \times \mathbb{Z}_N \mid \langle \mathbf{a}, \mathbf{e}_1^{(1)} \rangle \equiv s_1^{(1)} \bmod 2^{r^{(1)}} \right\},$$

$$L_2^{(1)} = \left\{ \left( \mathbf{e}_2^{(1)}, \langle \mathbf{a}, \mathbf{e}_2^{(1)} \rangle \right) \in D^n[0, 1/4] \times \mathbb{Z}_N \mid \langle \mathbf{a}, \mathbf{e}_2^{(1)} \rangle \equiv s_2^{(1)} \bmod 2^{r^{(1)}} \right\}.$$

where $s_2^{(1)} = s - s_1^{(1)}$. A collision where $\langle \mathbf{a}, \mathbf{e}_1^{(1)} \rangle = \langle \mathbf{a}, \mathbf{e}_2^{(1)} \rangle$, and $\mathbf{e}_1^{(1)} + \mathbf{e}_2^{(1)} \in D^n[0, 1/2]$ yields a subset-sum solution. If there are no such collisions, the process is repeated with a new $s_1^{(1)}$. After polynomially many iterations, the failure probability is negligible.

$L_1^{(1)}, L_2^{(1)}$ are constructed recursively, see also Figure 1. We demonstrate the list construction method using $L_1^{(1)}$ as an example. We represent $\mathbf{e}_1^{(1)} \in D^n[0, 1/4]$ as $\mathbf{e}_1^{(1)} = \mathbf{e}_1^{(2)} + \mathbf{e}_2^{(2)}$ with $\mathbf{e}_1^{(2)}, \mathbf{e}_2^{(2)} \in D^n[0, 1/8]$. By the same reasoning as before, the number of representations is $R^{(2)} = \binom{n/4}{n/8}$. Let $r^{(2)} = \lfloor \log_2 R^{(2)} \rfloor$. We compute

$$L_j^{(2)} = \left\{ \left( \mathbf{e}_j^{(2)}, \langle \mathbf{a}, \mathbf{e}_j^{(2)} \rangle \right) \in D^n[0, 1/8] \times \mathbb{Z}_N \mid \langle \mathbf{a}, \mathbf{e}_j^{(2)} \rangle \equiv s_j^{(2)} \bmod 2^{r^{(2)}} \right\},$$

for $j = 1, 2, 3, 4$, where $\sum_{j=1}^{4} s_j^{(2)} = s$.

The level-2 lists are built from the level-3 lists below via a standard Meet-in-the-Middle approach.

$$L_{2j-1}^{(3)} = \left\{ \left( \mathbf{e}_{2j-1}^{(3)}, \langle \mathbf{a}, \mathbf{e}_{2j-1}^{(3)} \rangle \right) \in D^{n/2}[0, 1/16] \times 0^{n/2} \times \mathbb{Z}_N \right\},$$

$$L_{2j}^{(3)} = \left\{ \left( \mathbf{e}_{2j}^{(3)}, \langle \mathbf{a}, \mathbf{e}_{2j}^{(3)} \rangle \right) \in 0^{n/2} \times D^{n/2}[0, 1/16] \times \mathbb{Z}_N \right\} \text{ for } j = 1, 2, 3, 4.$$

Algorithm 1 outlines the HGJ procedure. In summary, the algorithm recursively constructs two lists $L_1^{(1)}$ and $L_2^{(1)}$ that are expected to contain a single representation of $\mathbf{e}$. It iteratively performs a *merge-and-filter* operation across multiple levels, progressively approaching the distribution $D^n[0, 1/2]$ while incrementally increasing the bit-length of the modular constraint. This continues until the condition $\langle \mathbf{a}, \mathbf{e} \rangle \equiv s \pmod{2^n}$ is fully satisfied and a solution is found.

Consider two input distributions, $D_1 = D^n[\alpha_1, \beta_1]$ and $D_2 = D^n[\alpha_2, \beta_2]$, with a target distribution $D = D^n[\alpha, \beta]$. Given two lists $L_1 \in D_1^{|L_1|}$ and $L_2 \in D_2^{|L_2|}$, we define the following:

- The merged list

$$L_m = \{(\mathbf{e}_1 + \mathbf{e}_2, \langle \mathbf{a}, \mathbf{e}_1 + \mathbf{e}_2 \rangle) \mid \mathbf{e}_1 \in L_1, \mathbf{e}_2 \in L_2, \langle \mathbf{a}, \mathbf{e}_1 + \mathbf{e}_2 \rangle \equiv t \bmod M\},$$

where $0 \leq t < M$ is an arbitrary integer,

- The filtered list $L_f = (L_m \cap D) \subseteq L_m$, containing the vectors with the target distribution.

**Algorithm 1** The HGJ algorithm

**Input:** The random subset-sum instance $(\mathbf{a}, s) \in (\mathbb{Z}_N)^{n+1}$
**Output:** A solution $\mathbf{e} \in D^n[0, 1/2]$ or $\perp$ if no solution is found.

```
Construct all level-3 lists L_j^(3) for j ∈ {1, ..., 8}.
for i = 2 down to 0 do
    for j = 1 to 2^i do
        Construct L_j^(i) from L_{2j-1}^(i+1), L_{2j}^(i+1).
    end for
end for
if L_1^(0) ≠ ∅ then
    return any e ∈ L_1^(0)
else
    return ⊥
end if
```

A central tenet of the standard subset-sum heuristic is the treatment of $L_f$ elements as independent and uniformly distributed under $D$. The empirically validated heuristic allows for the derivation of classically provable, probabilistic algorithms, thereby corroborating the soundness of the heuristic itself (see Theorem 2 in [19]).

**Heuristic 1.** *Suppose the input vectors are uniformly distributed over $D_1 \times D_2$. Then the filtered pairs of vectors are also uniformly distributed over the target set $D$—or more precisely, over the subset of vectors in $D$ that satisfy the given modular constraint.*

By Heuristic 1, the expected size of the merged list is given by $|L_m| = (|L_1| \cdot |L_2|)/M$, and the average size of the filtered list $L_f$ equals $|L_m| \cdot \mathrm{pf}$, where pf denotes the probability that a pair $(\mathbf{e}_1, \mathbf{e}_2)$, sampled uniformly from $D_1 \times D_2$, satisfies $\mathbf{e}_1 + \mathbf{e}_2 \in D$.

For each level $i \in \{1, 2, 3\}$, the size of the corresponding lists is denoted by $L^{(i)}$. Under the same heuristic, the list sizes concentrate sharply around their expectations, leading to the expressions

$$L^{(3)} = \binom{n/2}{n/16}, L^{(2)} = \binom{n}{n/8}/R^{(2)}, L^{(1)} = \binom{n}{n/4}/R^{(1)}.$$

The time cost is

$$\max\left(L^{(3)}, \frac{\left(L^{(3)}\right)^2}{R^{(2)}}, \frac{\left(L^{(2)}\right)^2}{R^{(1)}/R^{(2)}}, \frac{\left(L^{(1)}\right)^2}{N/R^{(1)}}\right),$$

and the memory cost is $\max\left(L^{(3)}, L^{(2)}, L^{(1)}\right)$. Numerical optimization yields a time complexity of $\widetilde{\mathcal{O}}(2^{0.337n})$ and a space complexity of $\widetilde{\mathcal{O}}(2^{0.311n})$.

The algorithm of BCJ [19] achieves a time and space complexity of $\widetilde{\mathcal{O}}(2^{0.291n})$ by generalizing the representation technique to include entries from $\{-1, 0, 1\}$. In later work, Bonnetain et al. further improved this to $\widetilde{\mathcal{O}}(2^{0.283n})$ by relaxing BCJ constraints and allowing 2-coordinates in the representations.

In general, introducing more symbols ($-2$, 3, etc.) can increase the parameter search space and improve the optimal time complexity. However, the improvements gained by introducing further symbols are expected to exhibit diminishing returns.

To date, representation-based subset-sum algorithms in the literature exclusively employ binary trees as their search structure. In the following section, we propose new subset-sum algorithms based on ternary trees because ternary trees substantially increase the number of representations over binary trees.

## 3. A Novel Classical Subset-Sum Algorithm Based on Ternary Representation Techniques

### *3.1. Heuristic Construction*

The proposed classical algorithm, which we refer to as TER, employs a ternary tree structure as illustrated in Figure 2. We encourage the reader to consult the figure for a clearer understanding of the algorithmic workflow. The tree has a depth of 3—a value determined through numerical optimization to be optimal for TER (see Appendix A for details).

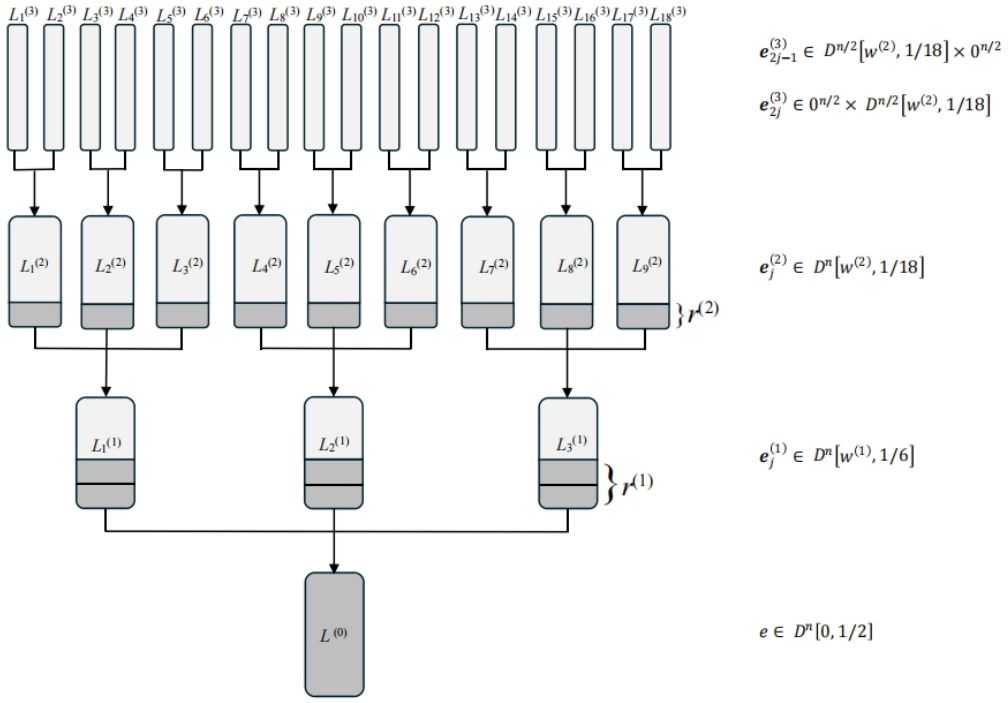


**Figure 2.** The ternary tree structure of TER. The darker regions indicate portions where the modulus constraint has been established.

For clarity, we consider a subset-sum instance $(\mathbf{a}, s) \in (\mathbb{Z}_N)^{n+1}$ with a solution $\mathbf{e} \in D^n[0, 1/2]$. The key idea is to represent $\mathbf{e}$ as a sum of three vectors in $D^n$, i.e.,

$$\mathbf{e} = \mathbf{e}_1^{(1)} + \mathbf{e}_2^{(1)} + \mathbf{e}_3^{(1)}$$

where $\mathbf{e}_1^{(1)}, \mathbf{e}_2^{(1)}, \mathbf{e}_3^{(1)} \in D^n[w^{(1)}, 1/6]$. This means each of these three vectors contains exactly $w^{(1)}n$ entries equal to $-1$, $(1/6 + w^{(1)})n$ entries equal to 1, and $(5/6 - 2w^{(1)})n$ entries equal to 0.

We begin by computing the number of representations and establishing the parameter $w^{(1)}$. For each 1-coordinate $e_k$ in $\mathbf{e}$, its decomposition via the $k$-th coordinates of $\mathbf{e}_1^{(1)}, \mathbf{e}_2^{(1)}, \mathbf{e}_3^{(1)}$ must be one of

$$1+0+0, 0+1+0, 0+0+1, 1+1+(-1), 1+(-1)+1, (-1)+1+1.$$

Let $\epsilon_1^{(1)} n$ denote the number of representations for each of the last three types. The total number of representations for the first three types is equal to $(1/2 - 3\epsilon_1^{(1)})n$, because $\mathbf{e}$ has $n/2$ 1-coordinates.

For each 0-coordinate $e_k$ in $\mathbf{e}$, its decomposition via the $k$-th coordinates of $\mathbf{e}_1^{(1)}, \mathbf{e}_2^{(1)}, \mathbf{e}_3^{(1)}$ must be one of

$$0+1+(-1), 0+(-1)+1, 1+0+(-1), (-1)+0+1, 1+(-1)+0, (-1)+1+0,$$

excluding the trivial case $0+0+0$. Let $\epsilon_0^{(1)} n$ denote the number of each of these six nontrivial representations.

Table 3 summarizes the counts for each type of level-1 representation.

**Table 3.** Count of different level-1 representations.

| | Representations | Count |
|---|---|---|
| 1-coordinates | $1+0+0$ | $(1/6-\epsilon_1^{(1)})n$ |
| | $0+1+0$ | $(1/6-\epsilon_1^{(1)})n$ |
| | $0+0+1$ | $(1/6-\epsilon_1^{(1)})n$ |
| | $1+1+(-1)$ | $\epsilon_1^{(1)} n$ |
| | $1+(-1)+1$ | $\epsilon_1^{(1)} n$ |
| | $(-1)+1+1$ | $\epsilon_1^{(1)} n$ |
| 0-coordinates | $0+1+(-1)$ | $\epsilon_0^{(1)} n$ |
| | $0+(-1)+1$ | $\epsilon_0^{(1)} n$ |
| | $1+0+(-1)$ | $\epsilon_0^{(1)} n$ |
| | $(-1)+0+1$ | $\epsilon_0^{(1)} n$ |
| | $1+(-1)+0$ | $\epsilon_0^{(1)} n$ |
| | $(-1)+1+0$ | $\epsilon_0^{(1)} n$ |
| | $0+0+0$ | $(1/2-6\epsilon_0^{(1)})n$ |

The total number of representations is given by

$$R^{(1)} = \binom{n/2}{\epsilon_1^{(1)} n, \epsilon_1^{(1)} n, \epsilon_1^{(1)} n, (1/6-\epsilon_1^{(1)})n, (1/6-\epsilon_1^{(1)})n, \cdot} \times \binom{n/2}{\epsilon_0^{(1)} n, \epsilon_0^{(1)} n, \epsilon_0^{(1)} n, \epsilon_0^{(1)} n, \epsilon_0^{(1)} n, \epsilon_0^{(1)} n, \cdot},$$

where the two factors, respectively, account for the number of representations for 1-coordinates and 0-coordinates in the vector $\mathbf{e}$.

The number of $(-1)$-coordinates of $\mathbf{e}_j^{(1)}$ for $j = 1, 2, 3$ is $(\epsilon_1^{(1)} + 2\epsilon_0^{(1)})n$, and the number of 1-coordinates is $(1/6 + \epsilon_1^{(1)} + 2\epsilon_0^{(1)})n$. Let $w^{(1)} = \epsilon_1^{(1)} + 2\epsilon_0^{(1)}$, then $\mathbf{e}_j^{(1)} \in D^n\left[w^{(1)}, 1/6\right]$ for $j = 1, 2, 3$.

To capture a representation of $\mathbf{e}$, we choose a modulus $2^{r^{(1)}}$ where $r^{(1)} = \lfloor \frac{1}{2} \log_2 R^{(1)} \rfloor$, and random targets $s_1^{(1)}, s_2^{(1)} \in \{0, 1, \ldots, 2^{r^{(1)}} - 1\}$. We then define the lists:

$$\begin{aligned} L_1^{(1)} &= \left\{ \left( \mathbf{e}_1^{(1)}, \langle \mathbf{a}, \mathbf{e}_1^{(1)} \rangle \right) \in D^n[w^{(1)}, 1/6] \times \mathbb{Z}_N \mid \langle \mathbf{a}, \mathbf{e}_1^{(1)} \rangle \equiv s_1^{(1)} \bmod 2^{r^{(1)}} \right\}, \\ L_2^{(1)} &= \left\{ \left( \mathbf{e}_2^{(1)}, \langle \mathbf{a}, \mathbf{e}_2^{(1)} \rangle \right) \in D^n[w^{(1)}, 1/6] \times \mathbb{Z}_N \mid \langle \mathbf{a}, \mathbf{e}_2^{(1)} \rangle \equiv s_2^{(1)} \bmod 2^{r^{(1)}} \right\}, \\ L_3^{(1)} &= \left\{ \left( \mathbf{e}_3^{(1)}, \langle \mathbf{a}, \mathbf{e}_3^{(1)} \rangle \right) \in D^n[w^{(1)}, 1/6] \times \mathbb{Z}_N \mid \langle \mathbf{a}, \mathbf{e}_3^{(1)} \rangle \equiv s_3^{(1)} \bmod 2^{r^{(1)}} \right\}, \end{aligned} \tag{1}$$

where $s_3^{(1)} = s - s_1^{(1)} - s_2^{(1)}$.

A valid solution is found when $\langle \mathbf{a}, \mathbf{e}_1^{(1)} \rangle + \langle \mathbf{a}, \mathbf{e}_2^{(1)} \rangle = \langle \mathbf{a}, \mathbf{e}_3^{(1)} \rangle$ and $\mathbf{e}_1^{(1)} + \mathbf{e}_2^{(1)} + \mathbf{e}_3^{(1)} \in D^n[0, 1/2]$. If no collision occurs, new values $(s_1^{(1)}, s_2^{(1)})$ are sampled. The failure probability becomes negligible after polynomially many attempts.

The expected size of each list $L_j^{(1)}$ $(j = 1, 2, 3)$ is

$$\mathbb{E}\left[|L_j^{(1)}|\right] = \frac{\binom{n}{w^{(1)}n, (1/6+w^{(1)})n, \cdot}}{\sqrt{R^{(1)}}}.$$

These lists are constructed recursively. For example, to build $L_1^{(1)}$, we represent $\mathbf{e}_1^{(1)} \in D^n[w^{(1)}, 1/6]$ as $\mathbf{e}_1^{(1)} = \mathbf{e}_1^{(2)} + \mathbf{e}_2^{(2)} + \mathbf{e}_3^{(2)}$ with $\mathbf{e}_1^{(2)}, \mathbf{e}_2^{(2)}, \mathbf{e}_3^{(2)} \in D^n[w^{(2)}, 1/18]$.

Let $\epsilon_1^{(2)} n$ count each of the representations $1+1+(-1), 1+(-1)+1$, and $(-1)+1+1$ for the 1-coordinates of $\mathbf{e}_1^{(1)}$, and $\epsilon_0^{(2)} n$ count each of the six nontrivial representations of the 0-coordinates. For the $(-1)$-coordinates, the representations are

$$(-1)+0+0, 0+(-1)+0, 0+0+(-1), 1+(-1)+(-1), (-1)+1+(-1), (-1)+(-1)+1.$$

Let $\epsilon_{-1}^{(2)} n$ count each of the last three types. Table 4 provides the counts for each type of level-2 representation.

**Table 4.** Count of different level-2 representations.

| | **Representations** | **Count** |
|---|---|---|
| 1-coordinates | $1+0+0$ | $(1/18 + w^{(1)}/3 - \epsilon_1^{(2)})n$ |
| | $0+1+0$ | $(1/18 + w^{(1)}/3 - \epsilon_1^{(2)})n$ |
| | $0+0+1$ | $(1/18 + w^{(1)}/3 - \epsilon_1^{(2)})n$ |
| | $1+1+(-1)$ | $\epsilon_1^{(2)} n$ |
| | $1+(-1)+1$ | $\epsilon_1^{(2)} n$ |
| | $(-1)+1+1$ | $\epsilon_1^{(2)} n$ |
| 0-coordinates | $0+1+(-1)$ | $\epsilon_0^{(2)} n$ |
| | $0+(-1)+1$ | $\epsilon_0^{(2)} n$ |
| | $1+0+(-1)$ | $\epsilon_0^{(2)} n$ |
| | $(-1)+0+1$ | $\epsilon_0^{(2)} n$ |
| | $1+(-1)+0$ | $\epsilon_0^{(2)} n$ |
| | $(-1)+1+0$ | $\epsilon_0^{(2)} n$ |
| | $0+0+0$ | $(5/6 - 2w^{(1)} - 6\epsilon_0^{(2)})n$ |
| $-1$-coordinates | $(-1)+0+0$ | $(w^{(1)}/3 - \epsilon_{-1}^{(2)})n$ |
| | $0+(-1)+0$ | $(w^{(1)}/3 - \epsilon_{-1}^{(2)})n$ |
| | $0+0+(-1)$ | $(w^{(1)}/3 - \epsilon_{-1}^{(2)})n$ |
| | $1+(-1)+(-1)$ | $\epsilon_{-1}^{(2)} n$ |
| | $(-1)+1+(-1)$ | $\epsilon_{-1}^{(2)} n$ |
| | $(-1)+(-1)+1$ | $\epsilon_{-1}^{(2)} n$ |

The number of representations at this level is

$$R^{(2)} = \binom{(1/6+w^{(1)})n}{\epsilon_1^{(2)}n, \epsilon_1^{(2)}n, \epsilon_1^{(2)}n, (1/18+w^{(1)}/3-\epsilon_1^{(2)})n, (1/18+w^{(1)}/3-\epsilon_1^{(2)})n, \cdot} \times \binom{(5/6-2w^{(1)})n}{\epsilon_0^{(2)}n, \epsilon_0^{(2)}n, \epsilon_0^{(2)}n, \epsilon_0^{(2)}n, \epsilon_0^{(2)}n, \epsilon_0^{(2)}n, \cdot} \times \binom{w^{(1)}n}{\epsilon_{-1}^{(2)}n, \epsilon_{-1}^{(2)}n, \epsilon_{-1}^{(2)}n, (w^{(1)}/3-\epsilon_{-1}^{(2)})n, (w^{(1)}/3-\epsilon_{-1}^{(2)})n, \cdot}.$$

The number of $(-1)$-coordinates of $\mathbf{e}_j^{(2)}$ for $j = 1, 2, 3$ is calculated as

$$\begin{aligned}&\left(w^{(1)}/3 - \epsilon_{-1}^{(2)} + 2\epsilon_{-1}^{(2)} + 2\epsilon_0^{(2)} + \epsilon_1^{(2)}\right)n \\ =&\left(\epsilon_1^{(1)}/3 + 2\epsilon_0^{(1)}/3 + \epsilon_{-1}^{(2)} + \epsilon_1^{(2)} + 2\epsilon_0^{(2)}\right)n,\end{aligned}$$

and the number of 1-coordinates of $\mathbf{e}_j^{(1)}$ for $j = 1, 2, 3$ is calculated as

$$\begin{aligned}&\left(1/18 + w^{(1)}/3 - \epsilon_1^{(2)} + 2\epsilon_1^{(2)} + 2\epsilon_0^{(2)} + \epsilon_{-1}^{(2)}\right)n \\ =&\left(1/18 + \epsilon_1^{(1)}/3 + 2\epsilon_0^{(1)}/3 + \epsilon_{-1}^{(2)} + \epsilon_1^{(2)} + 2\epsilon_0^{(2)}\right)n.\end{aligned}$$

Let $w^{(2)} = \epsilon_1^{(1)}/3 + 2\epsilon_0^{(1)}/3 + \epsilon_{-1}^{(2)} + \epsilon_1^{(2)} + 2\epsilon_0^{(2)}$, then $\mathbf{e}_j^{(2)} \in D^n[w^{(2)}, 1/18]$ for $j = 1, 2, 3$.

Let $r^{(2)} = \lfloor \frac{1}{2}\log_2 R^{(2)} \rfloor$ and choose random targets $s_j^{(2)} \in \{0, 1, \ldots, 2^{r^{(2)}} - 1\}$ for $j \in \{1, 2, 4, 5, 7, 8\}$. We define

$$L_j^{(2)} = \left\{ \left(\mathbf{e}_j^{(2)}, \langle \mathbf{a}, \mathbf{e}_j^{(2)} \rangle\right) \in D^n\left[w^{(2)}, 1/18\right] \times \mathbb{Z}_N \mid \langle \mathbf{a}, \mathbf{e}_j^{(2)} \rangle \equiv s_j^{(2)} \bmod 2^{r^{(2)}} \right\},$$

for $j \in \{1, \ldots, 9\}$, where $s_{3k}^{(2)} = s_k^{(1)} - s_{3k-2}^{(2)} - s_{3k-1}^{(2)}$ for $k = 1, 2, 3$.

The expected size of each $L_j^{(2)}$ ($j \in \{1, \ldots, 9\}$) is

$$\mathbb{E}\left[|L_j^{(2)}|\right] = \frac{\binom{n}{w^{(2)}n, (1/18+w^{(2)})n, \cdot}}{\sqrt{R^{(2)}}}.$$

The level-2 lists are built from the level-3 lists via a Meet-in-the-Middle approach.

$$\begin{aligned}L_{2j-1}^{(3)} &= \left\{ \left(\mathbf{e}_{2j-1}^{(3)}, \langle \mathbf{a}, \mathbf{e}_{2j-1}^{(3)} \rangle\right) \in D^{n/2}\left[w^{(2)}, 1/18\right] \times 0^{n/2} \times \mathbb{Z}_N \right\}, \\ L_{2j}^{(3)} &= \left\{ \left(\mathbf{e}_{2j}^{(3)}, \langle \mathbf{a}, \mathbf{e}_{2j}^{(3)} \rangle\right) \in 0^{n/2} \times D^{n/2}\left[w^{(2)}, 1/18\right] \times \mathbb{Z}_N \right\},\end{aligned}$$

for $j \in \{1, \ldots, 9\}$, each of size

$$\binom{n/2}{(w^{(2)}/2)n, (1/36+w^{(2)}/2)n, \cdot}.$$

Note that we employ the binary tree only for the top level, because the representation technique is not used in this level.

Algorithm 2 outlines the TER algorithm with a depth of 3. In summary, it recursively constructs three lists $L_1^{(1)}, L_2^{(1)}, L_3^{(1)}$ that are expected to contain a single representation of $\mathbf{e}$. Similar to the HGJ algorithm, this algorithm executes the merge-and-filter operation repeatedly over several levels.

Consider three input distributions, $D_1 = D^n[\alpha_1, \beta_1]$, $D_2 = D^n[\alpha_2, \beta_2]$ and $D_3 = D^n[\alpha_3, \beta_3]$, with a target distribution $D = D^n[\alpha, \beta]$. Given three lists $L_1 \in D_1^{|L_1|}$, $L_2 \in D_2^{|L_2|}$ and $L_3 \in D_2^{|L_3|}$, we define the following:

- The merged list

$$L_m = \{(\mathbf{e}_1 + \mathbf{e}_2 + \mathbf{e}_3, \langle \mathbf{a}, \mathbf{e}_1 + \mathbf{e}_2 + \mathbf{e}_3 \rangle) \mid \mathbf{e}_1 \in L_1, \mathbf{e}_2 \in L_2, \mathbf{e}_3 \in L_3, \langle \mathbf{a}, \mathbf{e}_1 + \mathbf{e}_2 + \mathbf{e}_3 \rangle \equiv t \bmod M\}, \quad (2)$$

  where $0 \leq t < M$ is an arbitrary integer,
- The filtered list $L_f = (L_m \cap D) \subseteq L_m$, containing the vectors with the target distribution.

**Algorithm 2** TER with a depth of 3

**Input:** The random subset-sum instance $(\mathbf{a}, s) \in (\mathbb{Z}_N)^{n+1}$

**Parameters:** $\epsilon_1^{(1)}, \epsilon_0^{(1)}, \epsilon_1^{(2)}, \epsilon_0^{(2)}, \epsilon_{-1}^{(2)}$ that need to be optimized.

**Output:** A solution $\mathbf{e} \in D^n[0, 1/2]$ or $\perp$ if no solution is found.

```
Construct all level-3 lists L_j^(3) for j ∈ {1, ..., 18}.
Compute L_j^(2) from L_{2j-1}^(3), L_{2j}^(3) for j ∈ {1, ..., 9}.
for i = 1 down to 0 do
    for j = 1 to 3^i do
        Construct L_j^(i) from L_{3j-2}^(i+1), L_{3j-1}^(i+1), L_{3j}^(i+1).
    end for
end for
if L_1^(0) ≠ ∅ then
    return any e ∈ L_1^(0)
else
    return ⊥
end if
```

Similar to the standard subset-sum assumption, our heuristic assumes that elements in $L_f$ are independently and uniformly sampled from the distribution $D$.

**Heuristic 2.** *Suppose the input vectors are uniformly distributed over $D_1 \times D_2 \times D_3$. Then the filtered pairs of vectors are also uniformly distributed over the target set $D$—or more precisely, over the subset of vectors in $D$ that satisfy the given modular constraint.*

By Heuristic 2, the expected sizes of the merged list is given by $|L_m| = (|L_1| \cdot |L_2| \cdot |L_3|)/M$, and the average size of the filtered list $L_f$ equals $|L_m| \cdot \mathrm{pf}$, where pf denotes the probability that a uniformly random triple $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$ from $D_1 \times D_2 \times D_3$ satisfies $(\mathbf{e}_1 + \mathbf{e}_2 + \mathbf{e}_3) \in D$.

**Theorem 1.** *Let $(\mathbf{a}, s) \in (\mathbb{Z}_N)^{n+1}$ be a random subset-sum instance whose solution is a vector $\mathbf{e} \in D^n[0, 1/2]$. Under Heuristic 2, Algorithm 2 finds $\mathbf{e}$ in time $\widetilde{\mathcal{O}}(2^{0.2400n})$ and space $\widetilde{\mathcal{O}}(2^{0.2221n})$.*

**Proof.** For each level $i \in \{1, 2, 3\}$, the size of the corresponding lists is denoted by $L^{(i)}$. Under Heuristic 2, the list sizes concentrate sharply around their expectations, leading to the expressions

$$L^{(1)} = \binom{n}{w^{(1)}n, (1/6 + w^{(1)})n, \cdot} / \sqrt{R^{(1)}},$$

$$L^{(2)} = \binom{n}{w^{(2)}n, (1/18 + w^{(2)})n, \cdot} / \sqrt{R^{(2)}},$$
$$L^{(3)} = \binom{n/2}{(w^{(2)}/2)n, (1/36 + w^{(2)}/2)n, \cdot}.$$

where

$$\begin{aligned}
w^{(1)} &= \epsilon_1^{(1)} + 2\epsilon_0^{(1)},\\
w^{(2)} &= \epsilon_1^{(1)}/3 + 2\epsilon_0^{(1)}/3 + \epsilon_{-1}^{(2)} + \epsilon_1^{(2)} + 2\epsilon_0^{(2)},\\
R^{(1)} &= \binom{n/2}{\epsilon_1^{(1)}n, \epsilon_1^{(1)}n, \epsilon_1^{(1)}n, (1/6 - \epsilon_1^{(1)})n, (1/6 - \epsilon_1^{(1)})n, \cdot}\\
&\quad \times \binom{n/2}{\epsilon_0^{(1)}n, \epsilon_0^{(1)}n, \epsilon_0^{(1)}n, \epsilon_0^{(1)}n, \epsilon_0^{(1)}n, \cdot},\\
R^{(2)} &= \binom{(1/6 + w^{(1)})n}{\epsilon_1^{(2)}n, \epsilon_1^{(2)}n, \epsilon_1^{(2)}n, (1/18 + w^{(1)}/3 - \epsilon_1^{(2)})n, (1/18 + w^{(1)}/3 - \epsilon_1^{(2)})n, \cdot}\\
&\quad \times \binom{(5/6 - 2w^{(1)})n}{\epsilon_0^{(2)}n, \epsilon_0^{(2)}n, \epsilon_0^{(2)}n, \epsilon_0^{(2)}n, \epsilon_0^{(2)}n, \cdot}\\
&\quad \times \binom{w^{(1)}n}{\epsilon_{-1}^{(2)}n, \epsilon_{-1}^{(2)}n, \epsilon_{-1}^{(2)}n, (w^{(1)}/3 - \epsilon_{-1}^{(2)})n, (w^{(1)}/3 - \epsilon_{-1}^{(2)})n, \cdot},
\end{aligned}$$

and $\epsilon_1^{(1)}, \epsilon_0^{(1)}, \epsilon_1^{(2)}, \epsilon_0^{(2)}, \epsilon_{-1}^{(2)}$ are parameters to be optimized.

Based on the preceding analysis, the time complexity of Algorithm 2 can be derived as follows:

$$\max\left(L^{(3)}, \frac{\left(L^{(3)}\right)^2}{\sqrt{R^{(2)}}}, \frac{\left(L^{(2)}\right)^3}{\sqrt{R^{(1)}/R^{(2)}}}, \frac{\left(L^{(1)}\right)^3}{N/\sqrt{R^{(1)}}}\right),$$

and the space complexity is $\max\left(L^{(3)}, L^{(2)}, L^{(1)}\right)$. Let us focus on the asymptotic exponent in $\log_2$ and relative to $n$. Denote $l^{(i)} = \log_2(L^{(i)})/n$ for $i = 1, 2, 3$. Recall that $r^{(1)} = \lfloor \frac{1}{2}\log_2 R^{(1)} \rfloor$ and $r^{(2)} = \lfloor \frac{1}{2}\log_2 R^{(2)} \rfloor$. Computing the time complexity essentially amounts to solving the following optimization problem, where the objective function is

$$\max\left(l^{(3)}, 2l^{(3)} - r^{(2)}/n, 3l^{(2)} - (r^{(1)} - r^{(2)})/n, 3l^{(1)} - (1 - r^{(1)}/n)\right),$$

subject to the following constraints:

$$\begin{aligned}
l^{(1)} &= H(1/6 + w^{(1)}, w^{(1)}) - r^{(1)}/n,\\
l^{(2)} &= H(1/18 + w^{(2)}, w^{(2)}) - r^{(2)}/n,\\
l^{(3)} &= H(1/18 + w^{(2)}, w^{(2)})/2,\\
w^{(1)} &= \epsilon_1^{(1)} + 2\epsilon_0^{(1)},\\
w^{(2)} &= \epsilon_1^{(1)}/3 + 2\epsilon_0^{(1)}/3 + \epsilon_{-1}^{(2)} + \epsilon_1^{(2)} + 2\epsilon_0^{(2)},\\
\frac{2r^{(1)}}{n} &= H(2\epsilon_1^{(1)}, 2\epsilon_1^{(1)}, 2\epsilon_1^{(1)}, 1/3 - 2\epsilon_1^{(1)}, 1/3 - 2\epsilon_1^{(1)}, \cdot)/2\\
&\quad + H(2\epsilon_0^{(1)}, 2\epsilon_0^{(1)}, 2\epsilon_0^{(1)}, 2\epsilon_0^{(1)}, 2\epsilon_0^{(1)}, 2\epsilon_0^{(1)}, \cdot)/2,\\
\frac{2r^{(2)}}{n} &= H\left(\frac{\epsilon_1^{(2)}}{1/6 + w^{(1)}}, \frac{\epsilon_1^{(2)}}{1/6 + w^{(1)}}, \frac{\epsilon_1^{(2)}}{1/6 + w^{(1)}}, 1/3 - \frac{\epsilon_1^{(2)}}{1/6 + w^{(1)}}, 1/3 - \frac{\epsilon_1^{(2)}}{1/6 + w^{(1)}}, \cdot\right)\\
&\quad \times (1/6 + w^{(1)})
\end{aligned}$$

$$+ H\left(\frac{\epsilon_0^{(2)}}{5/6-2w^{(1)}}, \frac{\epsilon_0^{(2)}}{5/6-2w^{(1)}}, \frac{\epsilon_0^{(2)}}{5/6-2w^{(1)}}, \frac{\epsilon_0^{(2)}}{5/6-2w^{(1)}}, \frac{\epsilon_0^{(2)}}{5/6-2w^{(1)}}, \frac{\epsilon_0^{(2)}}{5/6-2w^{(1)}}, \cdot\right)$$
$$\times (5/6-2w^{(1)})$$
$$+ H\left(\frac{\epsilon_{-1}^{(2)}}{w^{(1)}}, \frac{\epsilon_{-1}^{(2)}}{w^{(1)}}, \frac{\epsilon_{-1}^{(2)}}{w^{(1)}}, 1/3-\frac{\epsilon_{-1}^{(2)}}{w^{(1)}}, 1/3-\frac{\epsilon_{-1}^{(2)}}{w^{(1)}}, \cdot\right) \times w^{(1)}.$$

Note that Lemma 1 is applied when computing the exponents of the constraints. Numerical optimization yields the parameters

$$\epsilon_0^{(1)} = 0.0066, \epsilon_1^{(1)} = 0.0024, \epsilon_0^{(2)} = 0.0004, \epsilon_1^{(2)} = 0.0001, \epsilon_{-1}^{(2)} = 0.0000,$$

$$w^{(1)} = 0.0156, w^{(2)} = 0.0059.$$

This results in

$$R^{(1)} = 2^{1.1473n}, R^{(2)} = 2^{0.3396n},$$

which in turn yield list sizes

$$L^{(3)} = 2^{0.1922n}, L^{(2)} = 2^{0.2147n}, L^{(1)} = 2^{0.2221n}, L^{(0)} = 1.$$

The time complexity is $\widetilde{\mathcal{O}}\left(2^{0.2400n}\right)$, determined by the merged list size, and the space complexity is $\widetilde{\mathcal{O}}\left(2^{0.2221n}\right)$, governed by the filtered list size. □

### *3.2. Rigorous Analysis of TER*

In this subsection, we establish rigorous foundations for Heuristic 2 and present a provable variant of TER that eliminates heuristic assumptions. Our analysis relies on distributional properties of modular sums, as captured by the following fundamental result (Theorem 3.2 in [46]).

**Theorem 2.** *For any set $\mathcal{B} \subset \mathbb{Z}_M^n$, the identity*

$$\frac{1}{M^n} \sum_{\mathbf{a} \in \mathbb{Z}_M^n} \sum_{c \in \mathbb{Z}_M} \left(P_{\mathbf{a}}(\mathcal{B}, c) - \frac{1}{M}\right)^2 = \frac{M-1}{M|\mathcal{B}|} \tag{3}$$

*holds, where $P_{\mathbf{a}}(\mathcal{B}, c)$ denotes the probability that $\mathbf{a} \cdot \mathbf{x} := \langle \mathbf{a}, \mathbf{x} \rangle \equiv c \pmod{M}$ for a random $\mathbf{x}$ drawn uniformly from $\mathcal{B}$, i.e.,*

$$P_{\mathbf{a}}(\mathcal{B}, c) = \frac{1}{|\mathcal{B}|} |\{(x_1, \ldots, x_n) \in \mathcal{B} \text{ s.t. } \mathbf{a} \cdot \mathbf{x} \equiv c \bmod M\}|.$$

The left-hand side of Equation (3) represents the average squared deviation from uniformity across all coefficient vectors $\mathbf{a} = (a_1, \ldots, a_n)$ and residues $c \in \mathbb{Z}_M$. This quantity measures the overall non-uniformity of the distribution. The right-hand side being constant for fixed $|\mathcal{B}|$ implies that if certain subset-sum instances exhibit significant deviation from uniformity, others must be exceptionally uniform to maintain the average.

We now extend this result to the Cartesian product setting relevant to our decomposition approach.

**Theorem 3.** *For any sets $\mathcal{B}_1, \mathcal{B}_2 \subset \mathbb{Z}_M^n$ and the same coefficient vector $\mathbf{a} \in \mathbb{Z}_M^n$, the identity*

$$\frac{1}{M^n} \sum_{\mathbf{a} \in \mathbb{Z}_M^n} \sum_{(c_1, c_2) \in \mathbb{Z}_M^2} \left(P_{\mathbf{a}}(\mathcal{B}_1 \times \mathcal{B}_2, c_1, c_2) - \frac{1}{M^2}\right)^2 = \frac{(M-1)^2}{M^2|\mathcal{B}_1||\mathcal{B}_2|} \tag{4}$$

*holds, where* $P_{\mathbf{a}}(\mathcal{B}_1 \times \mathcal{B}_2, c_1, c_2)$ *denotes the probability that* $\mathbf{a} \cdot \mathbf{x} \equiv c_1 \pmod{M}$ *and* $\mathbf{a} \cdot \mathbf{y} \equiv c_2 \pmod{M}$ *for random* $(\mathbf{x}, \mathbf{y})$ *drawn uniformly from* $\mathcal{B}_1 \times \mathcal{B}_2$.

**Proof.** See Appendix C for the full proof. □

We now analyze pathological instances that cause TER to fail. During ternary decomposition (e.g., representing $\mathbf{e}$ as $\mathbf{e}_1^{(1)} + \mathbf{e}_2^{(1)} + \mathbf{e}_3^{(1)}$), certain coefficient vectors—such as $\mathbf{a} = (0, \ldots, 0)$—prevent constructing valid level-1 lists (Equation (1)) with random targets $s_1^{(1)}, s_2^{(1)}$ due to sparse residue coverage.

Let $\mathcal{B}_1$ and $\mathcal{B}_2$ denote the vector sets in the first two parts of the ternary decomposition. We quantify how many residue pairs $(s_1^{(1)}, s_2^{(1)}) \in \mathbb{Z}_M^2$ remain uncovered by $\mathcal{B}_1 \times \mathcal{B}_2$. For parameter $\Lambda > 0$, define an instance as *Type I bad* if either $\mathcal{B}_1$ covers fewer than $M/\Lambda$ residues or $\mathcal{B}_2$ covers fewer than $M/\Lambda$ residues.

For such bad instances, at least $(\Lambda - 1)M/\Lambda$ residues lead to zero probability: $P_{\mathbf{a}}(\mathcal{B}_1, s_1^{(1)}) = 0$ or $P_{\mathbf{a}}(\mathcal{B}_2, s_2^{(1)}) = 0$. We now establish a lower bound for the variance under bad instances. Without loss of generality, assume that $\mathcal{B}_1$ covers fewer than $M/\Lambda$ residues. For at least $(\Lambda - 1)M/\Lambda$ uncovered residues $s_1^{(1)}$,

$$\left(P_{\mathbf{a}}(\mathcal{B}_1, s_1^{(1)}) P_{\mathbf{a}}(\mathcal{B}_2, s_2^{(1)}) - \frac{1}{M^2}\right)^2 = \frac{1}{M^4}.$$

For the remaining at most $M/\Lambda$ covered residues $s_1^{(1)}$, we consider the minimal uniformity scenario: each such residue has a probability of at least $\Lambda/M$ (since the total probability sums to 1), and $P_{\mathbf{a}}(\mathcal{B}_2, s_2^{(1)})$ is uniform. Then,

$$\left(P_{\mathbf{a}}(\mathcal{B}_1, s_1^{(1)}) P_{\mathbf{a}}(\mathcal{B}_2, s_2^{(1)}) - \frac{1}{M^2}\right)^2 \geq \frac{(\Lambda - 1)^2}{M^4}.$$

Consequently, the lower bound for the variance is

$$\sum_{s_1^{(1)}, s_2^{(1)} \in \mathbb{Z}_M} \left(P_{\mathbf{a}}(\mathcal{B}_1, s_1^{(1)}) P_{\mathbf{a}}(\mathcal{B}_2, s_2^{(1)}) - \frac{1}{M^2}\right)^2 \geq \frac{(\Lambda - 1)M^2}{\Lambda} \cdot \frac{1}{M^4} + \frac{M^2}{\Lambda} \cdot \frac{(\Lambda - 1)^2}{M^4} = \frac{\Lambda - 1}{M^2}.$$

Let $N_{\mathrm{I}}$ denote the number of Type I bad instances. By Theorem 3,

$$N_{\mathrm{I}} \cdot \frac{\Lambda - 1}{M^2} \leq M^n \cdot \frac{(M-1)^2}{M^2 |\mathcal{B}_1||\mathcal{B}_2|}.$$

Since $|\mathcal{B}_1||\mathcal{B}_2| \approx M^2$, the fraction of Type I bad instances is

$$\frac{N_I}{M^n} \leq \frac{1}{\Lambda - 1}.$$

Another failure mode occurs when intermediate lists grow beyond expected sizes, violating the complexity analysis in Theorem 1. We now rigorously analyze list sizes during TER's execution.

Let $L_f$ be the filtered list in a merge-and-filter operation with distributions $D$. Let $\mathcal{B}_3$ denote all vectors with distribution $D$. We want $|L_f| \leq \Lambda |\mathcal{B}_3|/M$ for parameter $\Lambda$. Define *Type II bad* instances as those where more than $M/(2\Lambda)$ residues $c$ satisfy $P\mathbf{a}(\mathcal{B}_3, c) \geq \Lambda/M$.

By Theorem 2,

$$\frac{N_{\mathrm{II}}}{M^n} \cdot \frac{M}{2\Lambda} \cdot \frac{(\Lambda - 1)^2}{M^2} \leq \frac{M - 1}{M\mathcal{B}_3} \leq \frac{1}{\mathcal{B}_3},$$

where $N_{\mathrm{II}}$ counts Type II bad instances. Thus,

$$N_{\mathrm{II}} \leq \frac{2\Lambda}{(\Lambda-1)^2} \cdot \frac{M}{\mathcal{B}_3} \cdot M^n \leq \frac{2\Lambda}{(\Lambda-1)^2} \cdot M^n.$$

The fraction of Type II bad instances is at most $\Lambda/(\Lambda-1)^2$.

Now consider the merged list size. Let $L_m$ be the merged list in a merge-and-filter operation (definition in Equation (2)). Let $\mathcal{B}_4$ contain all sums $\mathbf{e}_1 + \mathbf{e}_2 + \mathbf{e}_3$ without modular constraints. Decompose $M = M_1 \cdot M_2$, where $M_1$ represents active moduli from input lists and $M_2$ is the supplementary constraint introduced during $L_m$ construction.

Let $\sigma \pmod M$ denote the target sum for elements in $L_m$, with $\sigma_L \pmod{M_1}$, $\sigma_M \pmod{M_1}$, $\sigma_R \pmod{M_1}$ representing partial sums from the three input lists, respectively, satisfying $\sigma_L + \sigma_M + \sigma_R \equiv \sigma \pmod{M_1}$.

The size of $L_m$ admits the bound

$$\begin{aligned} |L_m| &= \sum_{\substack{c_1,c_2,c_3 \in \mathbb{Z}_M \\ c_1+c_2+c_3 \equiv \sigma \pmod{M_1}}} (|\mathcal{B}_4| \cdot P_{\mathbf{a}}(\mathcal{B}_4, c_1)) \cdot (|\mathcal{B}_4| \cdot P_{\mathbf{a}}(\mathcal{B}, c_2)) \cdot (|\mathcal{B}_4| \cdot P_{\mathbf{a}}(\mathcal{B}_4, c_3)) \\ &\leq \left[ \prod_{i=1}^{3} \sum_{\substack{c_i \in \mathbb{Z}_M \\ c_i \equiv \sigma_i \pmod{M_1}}} (|\mathcal{B}_4| \cdot P_{\mathbf{a}}(\mathcal{B}_4, c_i))^3 \right]^{1/3}. \end{aligned}$$

To estimate this quantity, we require bounds on sums of the form

$$\sum_{\substack{c \in \mathbb{Z}_M \\ c \equiv c_0 \pmod{M_1}}} P_{\mathbf{a}}(\mathcal{B}, c)^3.$$

We can extend the relation from Theorem 2 to third moments:

$$\frac{1}{M^n} \sum_{\mathbf{a} \in \mathbb{Z}_M^n} \sum_{c \in \mathbb{Z}_M} P_{\mathbf{a}}(\mathcal{B}, c)^3 = \frac{M^2 + 3M|\mathcal{B}| - 3M + |\mathcal{B}|^2 - 3|\mathcal{B}| + 2}{M^2 |\mathcal{B}|^2}.$$

Define *Type III bad* instances as those where more than $M_1/(12\Lambda)$ values $c_0$ satisfy

$$\sum_{\substack{c \in \mathbb{Z}_M \\ c \equiv c_0 \pmod{M_1}}} P_{\mathbf{a}}(\mathcal{B}_4, c)^3 \geq \frac{\Lambda^2}{M_1^3 M_2}.$$

Letting $N_{\mathrm{III}}$ count such instances,

$$\frac{N_{\mathrm{III}}}{M^n} \cdot \frac{M_1}{12\Lambda} \cdot \frac{\Lambda^2}{M_1^3 M_2} \leq \frac{M^2 + 3M|\mathcal{B}_4| - 3M + |\mathcal{B}_4|^2 - 3|\mathcal{B}_4| + 2}{M^2 |\mathcal{B}_4|^2}.$$

which implies

$$N_{\mathrm{III}} \leq \frac{12}{\Lambda} \cdot \frac{M^2 + 3M|\mathcal{B}_4| + |\mathcal{B}_4|^2}{|\mathcal{B}_4|^2} \cdot M^n.$$

In TER, $|\mathcal{B}_4| \geq M$ for $L_m$ construction, so $F_{\mathrm{III}} \leq (48/\Lambda) M^n$. The fraction of Type III bad instances is at most $48/\Lambda$.

This analysis provides theoretical guarantees for all intermediate lists in TER. Since depth-3 TER uses four ternary decompositions, the total fraction of bad instances is bounded by

$$4\left( \frac{1}{\Lambda - 1} + \frac{2\Lambda}{(\Lambda-1)^2} + \frac{48}{\Lambda} \right) \leq \frac{212}{\Lambda}.$$

By choosing sufficiently large $\Lambda$, this fraction becomes negligible.

Excluding three types of bad subset-sum instances, we analyze the success probability of algorithm TER on the remaining good instances, where modulus values are randomly resampled in each ternary decomposition. For good instances, failures fall into three categories:

1. Selecting residues that yield zero probability. With at most $(\Lambda-1)M/\Lambda$ such residues, the failure probability is $(\Lambda-1)/\Lambda$.
2. The filtered list overflow. With at most $M/(2\Lambda)$ residues, the failure probability is $(\Lambda-1)/\Lambda$.
3. The merged list overflow. For each of the three lists to be merged, at most $M_1/(12\Lambda)$ residues cause overflow, and the failure probability here is $3/(12\Lambda)$.

Thus, the overall failure probability is

$$\frac{\Lambda-1}{\Lambda}+\frac{1}{2\Lambda}+\frac{3}{12\Lambda}=1-\frac{1}{4\Lambda}.$$

Using the limit formula $\lim_{n\to\infty}(1-1/n)^n = e^{-1}$, we repeat each ternary decomposition $8\Lambda$ times to ensure a failure probability of at most $e^{-2}$. Since $e^{-2} \approx 0.135$, and TER involves four ternary decompositions, the overall failure probability is at most $4\times 0.135 = 0.54$. Hence, the success probability is at least 0.46. If this success probability is insufficient, a polynomial number of repetitions yields a success probability exponentially close to 1, excluding the bad subset-sum instances. Therefore, this repetition strategy allows us to construct a provable probabilistic version of TER.

## 4. Improved Subset-Sum Algorithm Based on Quantum Walks

In this section, we begin by introducing the MNRS framework for searching marked vertices on graphs using quantum walks. We then describe how our classical algorithm, TER, is incorporated into this quantum walk framework. Subsequently, we provide a detailed complexity analysis of our quantum algorithm. Through numerical optimization, we derive the optimal parameters and their corresponding complexity values.

### *4.1. Quantum Walks*

Consider an undirected graph $G=(V,E)$ that is both connected and regular. Suppose that some vertices of $G$ are "marked." We denote by $\epsilon$ the fraction of marked vertices; equivalently, a vertex chosen uniformly at random is marked with probability $\epsilon$. Furthermore, let $\delta$ be the spectral gap of $G$, i.e., the difference between its two largest eigenvalues. For a classical random walk on $G$ starting from any vertex, the number of steps required to approximate the stationary distribution is roughly $1/\delta$.

A classical random walk searches for a marked vertex via the following subroutines:

1. **Setup.** Sample a random vertex $v \in V$ in time $T_S$.
2. **Update.** Perform $1/\delta$ random walk steps, where each step moves to a uniformly random adjacent vertex in time $T_U$.
3. **Checking.** Check whether the current vertex is marked in time $T_C$. If not, return to Step 2.

The total time complexity is

$$T_{\mathrm{RW}} = T_S + \frac{1}{\epsilon}\left(\frac{1}{\delta}T_U + T_C\right).$$

In the quantum setting, the walk takes place over a superposition of vertices. The initial state is the uniform superposition $\sum_{v\in V}|v\rangle$. The algorithm proceeds by iterating

approximately $\sqrt{1/\epsilon}$ times, each iteration amplifying the amplitude on marked vertices. Crucially, each such iteration requires only about $\sqrt{1/\delta}$ update steps to sufficiently mix the quantum state. An update step in the quantum walk maps a vertex to a superposition of its neighbors. The MNRS framework [32] enables us to focus on specifying the setup, checking, and update unitaries.

**Theorem 4** (Quantum walk on a graph [32]). *Given a regular graph $G = (V, E)$ with spectral gap $\delta > 0$, where at least an $\epsilon > 0$ fraction of the vertices are marked, for a random walk on a graph $G$ with setup, update, and checking costs denoted by $T_S$, $T_U$, and $T_C$, respectively, there exists a quantum algorithm that can find a marked vertex with high probability in time*

$$T_{QW} = \mathcal{O}\left(T_S + \frac{1}{\sqrt{\epsilon}}\left(T_C + \frac{T_U}{\sqrt{\delta}}\right)\right). \tag{5}$$

Johnson graphs are central to the construction of our quantum walk.

**Definition 3** (Johnson graph). *The Johnson graph, denoted $J(N,k)$, is an undirected graph whose vertices correspond to all $k$-element subsets of a universe of size $N$. Two vertices $S$ and $S'$ are connected by an edge if and only if $|S \cap S'| = k-1$. This means $S'$ can be obtained from $S$ by replacing exactly one element. The spectral gap of $J(N,k)$ is given by*

$$\delta(J(N,k)) = \frac{N}{k(N-k)}.$$

**Definition 4** (Cartesian product of Johnson graphs). *Let $J^m(N,k)$ denote the Cartesian product of $m$ copies of the Johnson graph $J(N,k)$. Each vertex in $J^m(N,k)$ is an $m$-tuple $(S_1, \ldots, S_m)$ of $k$-element subsets. Two vertices $(S_1, \ldots, S_m)$ and $(S'_1, \ldots, S'_m)$ are adjacent if and only if they differ in exactly one coordinate $i$, and for that index, $|S_i \cap S'_i| = k-1$. The spectral gap satisfies*

$$\delta(J^m(N,k)) \geq \frac{1}{m} \cdot \delta(J(N,k)) = \Omega\left(\frac{1}{k}\right). \tag{6}$$

**Remark 1** (Heuristics on quantum walks [26]). *A notable distinction between classical and quantum subroutines is that quantum updates are required to terminate within a specified time limit, whereas classical procedures may run indefinitely in worst-case scenarios, albeit with low probability. This issue was studied in (Section 6 in [47]) and (Section 5 in [20]). It was shown that terminating updates within a polynomial factor of their expected runtime will not significantly impact final quantum states. Hence, up to a polynomial overhead, it is sufficient to analyze the quantum walk complexity using the expected costs.*

### *4.2. Quantum TER (QTER)*

Next, we present the quantum version of our classical algorithm TER, termed QTER. Recall the tree structure illustrated in Figure 2. The depth of the tree in our quantum algorithm is also a parameter subject to optimization, and a depth of 3 was found to be optimal for QTER (as detailed in the Appendix A).

In the quantum walk setup, we construct random subsets $U_j^{(3)} \subseteq L_j^{(3)}$ for $j \in \{1, \ldots, 18\}$, each of size $U^{(3)} := (L^{(3)})^\gamma$ with $\gamma < 1$. The TER algorithm is then executed using these newly formed depth-3 lists $U_j^{(3)}$.

Let $N = L^{(3)}$ and $k = U^{(3)}$. For each list $L_j^{(3)}$, where $j \in \{1, \ldots, 18\}$, we associate a Johnson graph denoted $J_j(N,k)$. The overall structure for the random walk is defined on the Cartesian product graph

$$J(N,k) = J_1(N,k) \times J_2(N,k) \times \cdots \times J_{18}(N,k).$$

The vertices of $J(N,k)$ are $(U_1^{(3)}, U_2^{(3)}, \ldots, U_{18}^{(3)})$, with $U_j^{(3)} \subseteq L_j^{(3)}, |U_j^{(3)}| = k, 1 \leq j \leq 18$. Denote by $U_j^{(i)}$ the $j$-th list at level $i$, which is constructed from $U_1^{(3)}, \ldots, U_{18}^{(3)}$ according to TER, for $0 \leq i \leq 2, 1 \leq j \leq 3^i$. The data structure of a vertex in the graph $J(N,k)$ maintains all such lists $U_j^{(i)}$. A vertex is considered marked if $U_1^{(0)}$ includes a solution to the initial random subset-sum problem. It can be seen that solving the random subset-sum problem can be transformed into the problem of finding a marked vertex on a graph.

The pseudocode of QTER with depth 3 is given in Algorithm 3.

We now analyze the quantum resources required by Algorithm 3. The quantum circuit width, measured in qubits, comprises three main components: the current vertex registers $\bigotimes_{j=1}^{18} |U_j^{(3)}\rangle$, the coin register $|coin\rangle$, and the data register $|data\rangle$.

The tensor product of 18 vertex registers requires $18U^{(3)}$ qubits. The coin register, which stores superpositions of neighboring vertices, also requires $18U^{(3)}$ qubits. The data register stores all hierarchical lists constructed from the current vertex, namely, the level-2, level-1, and level-0 lists, requiring $9U^{(2)} + 3U^{(1)} + U^{(0)}$ qubits in total, where $U^{(i)} (i = 0, 1, 2)$ denotes the size of each level-$i$ list. Since the list sizes $U^{(i)}$ grow exponentially with the problem parameters, the overall circuit width is dominated by $\max(U^{(3)}, U^{(2)}, U^{(1)}, U^{(0)})$ when ignoring polynomial factors.

The circuit depth is determined by three key subroutines. Let $T_S$, $T_U$, and $T_C$ represent the circuit depths of the setup, update, and checking operations, respectively. The total circuit depth of Algorithm 3 is given by

$$T_S + \frac{1}{\sqrt{\epsilon}} \left( T_C + \frac{T_U}{\sqrt{\delta}} \right),$$

which is consistent with Theorem 4. The complexity of QTER therefore depends on the parameters $T_S$, $T_U$, $T_C$, $\epsilon$, and $\delta$.

The checking subroutine achieves $T_C = \widetilde{\mathcal{O}}(1)$ through the following procedure: We allocate one ancilla qubit and apply a controlled operation using the register storing $U_1^{(0)}$ as control and the ancilla as target. This operation flips the ancilla qubit if and only if $U_1^{(0)}$ contains a valid solution. The ancilla qubit then facilitates the phase flip for marked vertices. Finally, we uncompute the ancilla by applying the conjugate transpose of the controlled operation. This entire process requires only a polynomial number of quantum gates and one ancilla qubit, ensuring $T_C = \widetilde{\mathcal{O}}(1)$.

We will analyze $T_S$, $T_U$, $\epsilon$, and $\delta$ in the next subsection to determine the overall complexity of QTER.

**Algorithm 3** QTER with depth 3

**Input:** The random subset-sum instance $(\mathbf{a}, s) \in (\mathbb{Z}_N)^{n+1}$

**Parameters:** $\gamma, \epsilon_1^{(1)}, \epsilon_0^{(1)}, \epsilon_1^{(2)}, \epsilon_0^{(2)}, \epsilon_{-1}^{(2)}$ that need to be optimized.

**Output:** A solution $\mathbf{e} \in D^n\left[0, 1/2\right]$ or $\perp$ if no solution is found.

1: [Setup] Prepare the initial state (normalization omitted):

$$\sum_{(U_1^{(3)} \times \cdots \times U_{18}^{(3)}) \in J(N,k)} \bigotimes_{j=1}^{18} |U_j^{(3)}\rangle \, |coin\rangle \, |data\rangle |0\rangle,$$

where $U_j^{(3)} \subseteq L_j^{(3)}$ of size $(L^{(3)})^\gamma$ for $j \in \{1, \ldots, 18\}$, the $|coin\rangle$ register represents a superposition of the neighboring vertices, and the $|data\rangle$ register stores the data associated with the current vertex. ▷ Parameters $\epsilon_1^{(1)}, \epsilon_0^{(1)}, \epsilon_1^{(2)}, \epsilon_0^{(2)}, \epsilon_{-1}^{(2)}$ are employed in the construction of the $|data\rangle$ state.

2: **Repeat** $O(1/\sqrt{\epsilon})$ times:

(2.1) [Checking]

$$\bigotimes_{j=1}^{18} |U_j^{(3)}\rangle \, |coin\rangle \, |data\rangle \mapsto \begin{cases} -\bigotimes_{j=1}^{18} |U_j^{(3)}\rangle \, |coin\rangle \, |data\rangle, & \text{if the vertex is marked,} \\ \bigotimes_{j=1}^{18} |U_j^{(3)}\rangle \, |coin\rangle \, |data\rangle, & \text{else.} \end{cases}$$

(2.2) [Update] **Repeat** $O(1/\sqrt{\delta})$ times:

(2.2.1) Perform a single step of the quantum walk.

(2.2.2) Update the data structure.

3: Measure the register $|data\rangle$.

4: **if** $U_1^{(0)} \neq \emptyset$ **then**

5: **return** any $\mathbf{e} \in U_1^{(0)}$

6: **else**

7: **return** $\perp$

8: **end if**

### *4.3. The Complexity of QTER*

From Equation (6), the spectral gap of $J(N, k)$ is

$$\delta(J(N,k)) = \Omega\left(\frac{1}{k}\right) = \Omega\left(\frac{1}{U^{(3)}}\right). \tag{7}$$

A node $v$ that is marked equivalent to its associated level-0 list $U_1^{(0)}$ is non-empty; that is, it retains at least one representation of $\mathbf{e}$ that survives all merge-and-filter operations. Note that the parameter selection for the lists $L_j^{(3)}$ within the TER framework ensures that, in expectation, there exists a representation

$$\mathbf{e} = \mathbf{e}_1^{(3)} + \cdots + \mathbf{e}_{18}^{(3)}$$

which persists through all filtering steps up to the root list $L_1^{(0)}$. Under the QTER construction, the probability that $\mathbf{e}_j^{(3)} \in U_j^{(3)}$ holds simultaneously for all $j \in \{1, \ldots, 18\}$ is given by

$$\epsilon = \left(\frac{U^{(3)}}{L^{(3)}}\right)^{18} = \left(L^{(d)}\right)^{18(\gamma-1)}. \tag{8}$$

Hence, the probability that $v$ is marked is $\epsilon$.

As shown in Section 3, for TER we obtain the list sizes for each level:

$$\begin{aligned} L^{(1)} &= \binom{n}{w^{(1)}n, (1/6 + w^{(1)})n, \cdot} / \sqrt{R^{(1)}}, \\ L^{(2)} &= \binom{n}{w^{(2)}n, (1/18 + w^{(2)})n, \cdot} / \sqrt{R^{(2)}}, \\ L^{(3)} &= \binom{n/2}{(w^{(2)}/2)n, (1/36 + w^{(2)}/2)n, \cdot}. \end{aligned}$$

We first calculate the setup cost $T_S$ of QTER with depth 3. The size of the QTER level-3 list is

$$U^{(3)} = \left(L^{(3)}\right)^{\gamma} = \binom{n/2}{(w^{(2)}/2)n, (1/36 + w^{(2)}/2)n, \cdot}^{\gamma}.$$

The level-3 lists $U_j^{(3)}$ can be computed and sorted with respect to the inner products $\langle \mathbf{a}, \mathbf{e}_j^{(3)} \rangle$ in time $U^{(3)}$ for $j \in \{1, \ldots, 18\}$. The level-2 lists contain all elements from their two level-3 children lists that match on the inner products. Thus, we expect $U^{(2)} = (U^{(3)})^2 / \sqrt{R^{(2)}}$ elements that match on their inner products.

Analogously, we compute level-1 lists in expected time $(U^{(2)})^3 \sqrt{R^{(2)}} / \sqrt{R^{(1)}}$. However, now we have to filter out all $\mathbf{e}_j^{(1)} \notin D^n[w^{(1)}, 1/6]$ for $j = 1, 2, 3$.

Denote $pr^{(i)}$ as the probability that a uniformly random triple $(\mathbf{e}_1^{(i+1)}, \mathbf{e}_2^{(i+1)}, \mathbf{e}_3^{(i+1)}) \in L_1^{(i+1)} \times L_2^{(i+1)} \times L_3^{(i+1)}$ from $(D^n[w^{(i+1)}, 1/(2 \times 3^{i+1})])^3$ satisfies $(\mathbf{e}_1^{(i)} + \mathbf{e}_2^{(i)} + \mathbf{e}_3^{(i)}) \in D^n[w^{(i)}, 1/(2 \times 3^i)]$, for $i = 0, 1$. The specific calculation formula for $pr^{(0)}, pr^{(1)}$ can be found in Appendix B.

By leveraging the filtering probability, the size of level-1 lists can be estimated as follows:

$$U^{(1)} = \frac{\left(U^{(2)}\right)^3}{\sqrt{R^{(1)}/R^{(2)}}} \cdot pr^{(1)}.$$

The level-0 list can be computed in expected time $(U^{(1)})^3 \sqrt{R^{(1)}}/N$ and

$$U^{(0)} = \frac{\left(U^{(1)}\right)^3}{N/\sqrt{R^{(1)}}} \cdot pr^{(0)}.$$

Thus, the setup cost can be bounded as

$$\begin{aligned} T_S &= \max\left\{ U^{(3)}, \frac{\left(U^{(3)}\right)^2}{\sqrt{R^{(2)}}}, \frac{\left(U^{(2)}\right)^3}{\sqrt{R^{(1)}/R^{(2)}}}, \frac{\left(U^{(1)}\right)^3}{N/\sqrt{R^{(1)}}} \right\} \\ &\leq \max\left\{ U^{(3)}, \frac{\left(U^{(3)}\right)^2}{\sqrt{R^{(2)}}}, \frac{\left(U^{(3)}\right)^6}{\sqrt{R^{(1)}}R^{(2)}}, \frac{\left(U^{(3)}\right)^{18}}{NR^{(1)}\left(R^{(2)}\right)^3} \right\}. \end{aligned}$$

The update procedure involves inserting and deleting an element from one of the sets $U_j^{(3)}$, where $j \in \{1, \ldots, 18\}$. For instance, suppose we wish to replace an element $e_4^{(3)}$ in $U_4^{(3)}$. This update operation on $U_4^{(3)}$ can be performed in constant time, i.e., in $\mathcal{O}(1)$.

This update affects the lower-level list $U_2^{(2)}$ if there exist matching elements $\mathbf{e}_3^{(3)}$ such that $\langle \mathbf{a}, \mathbf{e}_3^{(3)} + \mathbf{e}_4^{(3)} \rangle = s_2^{(2)} \bmod 2^{r^{(2)}}$. The expected number of such elements is $U^{(3)}/\sqrt{R^{(2)}}$, applicable to both deletion and insertion operations. At level 1, for the list $U_1^{(1)}$, each of the $U^{(3)}/\sqrt{R^{(2)}}$ elements leads to an expected number of $(U^{(2)})^2\sqrt{R^{(2)}}/\sqrt{R^{(1)}}$ insertions or deletions. A similar reasoning applies to the level-0 list $U_1^{(0)}$. Overall, the total expected cost of an update is given by

$$\begin{aligned} T_U &= \max\left\{1, \frac{U^{(3)}}{\sqrt{R^{(2)}}}, \frac{\left(U^{(2)}\right)^2 U^{(3)}}{\sqrt{R^{(1)}}}, \frac{\left(U^{(1)}U^{(2)}\right)^2 U^{(3)}}{N}\right\} \\ &\leq \max\left\{1, \frac{U^{(3)}}{\sqrt{R^{(2)}}}, \frac{\left(U^{(3)}\right)^5}{\sqrt{R^{(1)}}R^{(2)}}, \frac{\left(U^{(3)}\right)^{17}}{NR^{(1)}\left(R^{(2)}\right)^3}\right\}. \end{aligned}$$

Observe that if we approximate the values of $T_S$ and $T_U$ using their upper bounds, the relation $T_S = U^{(3)}T_U$ holds. By combining Equations (5), (7) and (8) with $T_C = \widetilde{\mathcal{O}}(1)$, we derive the following expression for the quantum walk runtime:

$$\begin{aligned} T_{QW} &\leq T_S + \frac{1}{\sqrt{\epsilon}}\left(\frac{1}{\sqrt{\delta}}T_U + T_C\right) \\ &= U^{(3)}T_U + \left(\frac{L^{(3)}}{U^{(3)}}\right)^9\left(\sqrt{U^{(3)}}T_U + 1\right) \\ &= \left(L^{(3)}\right)^{\gamma}T_U + \left(L^{(3)}\right)^{9(1-\gamma)+\frac{\gamma}{2}}T_U \\ &= 2\left(L^{(3)}\right)^{\frac{18}{19}}T_U. \end{aligned}$$

To balance setup costs with the cost to find a marked node, we set parameter $\gamma$ to 18/19. This value satisfies the equation $\gamma = 9(1-\gamma) + \frac{\gamma}{2}$. If $\gamma$ is greater than 18/19, the setup costs increase, leading to a larger overall complexity $T_{QW}$; if $\gamma$ is smaller than 18/19, the cost of the update and checking iterations becomes higher, also resulting in an increase in the overall complexity $T_{QW}$.

It should be noted that the exponent $\gamma$ converges to 1 as the tree depth increases. This implies that QTER degrades to TER as the number of the depth increases. For the QTER algorithm, a depth of 3 is found to be optimal.

Analogously to Section 3, we obtain the following parameters through numerical optimization:

$$\epsilon_0^{(1)} = 0.0057, \epsilon_1^{(1)} = 0.0020, \epsilon_0^{(2)} = 0.0009, \epsilon_1^{(2)} = 0.0001, \epsilon_{-1}^{(2)} = 0.0000,$$

$$w^{(1)} = 0.0133, w^{(2)} = 0.0063.$$

This results in

$$R^{(1)} = 2^{1.1054n}, R^{(2)} = 2^{0.3685n},$$

which in turn yield list sizes

$$U^{(3)} = 2^{0.1843n}, U^{(2)} = 2^{0.1843n}, U^{(1)} = 2^{0.1639n}, U^{(0)} = 2^{-0.1230n}.$$

The time and space complexity are both $\widetilde{\mathcal{O}}\left(2^{0.1843n}\right)$.

**Theorem 5.** *Let* $(\mathbf{a}, s) \in (\mathbb{Z}_N)^{n+1}$ *be a random subset-sum instance whose solution is a vector* $\mathbf{e} \in D^n\left[0, 1/2\right]$. *Under the assumption of Heuristic 2, there exists a quantum algorithm (Algorithm 3) that finds* $\mathbf{e}$ *in time and space* $\widetilde{O}\left(2^{0.1843n}\right)$.

## 5. Application to Decoding

We now turn to the application of our subset-sum techniques to the problem of decoding random binary linear codes. This is a central problem in coding theory and cryptography, and it can be formulated as a subset-sum problem over the additive group $(\mathbb{F}_2^m, +)$ rather than $(\mathbb{Z}_{2^n}, +)$.

**Definition 5** (Syndrome decoding problem)**.** *Let* $P \in \mathbb{F}_2^{(n-k)\times n}$ *be a uniformly random full-rank matrix, which serves as the parity-check matrix of a random linear code* $\mathcal{C} = \{\mathbf{c} \in \mathbb{F}_2^n \mid P\mathbf{c} = \mathbf{0}\}$. *Let* $\mathbf{y} \in \mathbb{F}_2^n$ *be a received erroneous codeword. The decoding problem is to find an error vector* $\mathbf{e} \in \mathbb{F}_2^n$ *of minimal Hamming weight such that*

$$P\mathbf{e} = P\mathbf{y}.$$

*We denote the syndrome by* $\mathbf{s} = P\mathbf{y}$. *The goal is to recover* $\mathbf{e}$ *(and hence the codeword* $\mathbf{c} = \mathbf{y} + \mathbf{e}$*).*

Note that if $\mathbf{e}$ has weight at most $\left\lfloor \frac{d-1}{2} \right\rfloor$, where $d$ is the minimum distance of $\mathcal{C}$, then the solution is unique. This is referred to as the *half-distance decoding* scenario. If $|\mathbf{e}| \leq d$, we are in the *full-distance decoding* setting.

It i s well-known that the relative distance $d/n$ of a random linear code asymptotically meets the Gilbert–Varshamov bound:

$$\frac{d}{n} \approx H^{-1}\left(1 - \frac{k}{n}\right),$$

where $H^{-1}$ is the inverse of the binary entropy function. Thus, the complexity of decoding can be expressed as a function of the code rate $k/n$.

A key observation is that the condition $P\mathbf{e} = \mathbf{s}$ can be viewed as a subset-sum instance over $\mathbb{F}_2^{n-k}$: the columns of $P$ form the set of elements, and $\mathbf{s}$ is the target. The additional constraint is that the solution $\mathbf{e}$ must be a binary vector of low Hamming weight.

### *5.1. Information Set Decoding*

The core idea of ISD is to reduce the problem dimension via Gaussian elimination.

Let $\Pi$ be an $n \times n$ permutation matrix over $\mathbb{F}_2$. For any permutation matrix $\Pi$, $P\Pi^{-1} \cdot \Pi\mathbf{e} = \mathbf{s}$ holds. Let $\ell$ be a parameter such that $0 \leq \ell \leq n - k$. For any $\ell$, with constant probability, Gaussian elimination produces an invertible matrix $G \in \mathbb{F}_2^{(n-k)\times(n-k)}$ such that

$$G(P\Pi^{-1}) = \begin{bmatrix} P_1 & \mathbf{0} \\ P_2 & I_{n-k-\ell} \end{bmatrix},$$

where $P_1 \in \mathbb{F}_2^{\ell\times(k+\ell)}$, $P_2 \in \mathbb{F}_2^{(n-k-\ell)\times(k+\ell)}$, and $I_{n-k-\ell}$ is the identity matrix of size $n - k - \ell$.

Partition $\Pi\mathbf{e} = (\mathbf{e}', \mathbf{e}'')$ with $\mathbf{e}' \in \mathbb{F}_2^{k+\ell}$, $\mathbf{e}'' \in \mathbb{F}_2^{n-k-\ell}$, and similarly let $G\mathbf{s} = (\mathbf{s}', \mathbf{s}'')$. Then the equation $G(P\Pi^{-1})(\Pi\mathbf{e}) = G\mathbf{s}$ becomes

$$\begin{cases} P_1\mathbf{e}' = \mathbf{s}', \\ P_2\mathbf{e}' + \mathbf{e}'' = \mathbf{s}''. \end{cases}$$

Let us introduce an additional parameter $p$. We say that a permutation $\Pi$ is good if it induces a weight distribution satisfying $|\mathbf{e}'| = p$ and $|\mathbf{e}''| = \omega - p$. Assume such a good permutation is found, then the decoding problem reduces to finding a vector $\mathbf{e}' \in \mathbb{F}_2^{k+\ell}$ of weight $p$ such that $P_1\mathbf{e}' = \mathbf{s}'$, and $|P_2\mathbf{e}' + \mathbf{s}''| = \omega - p$. The complete solution is then given by $\mathbf{e} = \Pi^{-1}(\mathbf{e}', P_2\mathbf{e}' + \mathbf{s}'')$.

The probability that a randomly chosen permutation is good is

$$\mathcal{P}(p, \ell) = \frac{\binom{k+\ell}{p}\binom{n-k-\ell}{\omega-p}}{\binom{n}{\omega}}.$$

In the following subsection, we show how to solve the resulting subset-sum instance $P_1\mathbf{e}' = \mathbf{s}'$ efficiently using a variant of TER, adapted to the binary group setting.

### *5.2. TER for Decoding*

Let us focus on the subset-sum instance $\left[\mathbf{P_{1,1}} \| \mathbf{P_{1,2}} \| \cdots \| \mathbf{P_{1,k+\ell}} \| \mathbf{s}'\right]$ with solution $\mathbf{e}' \in \mathbb{F}_2^{k+\ell}$, where $\mathbf{P_{1,j}} (1 \leq j \leq k+\ell)$ stands for the $j$th column of matrix $P_1$ and $\mathbf{P_{1,1}}, \ldots, \mathbf{P_{1,k+\ell}}, \mathbf{s}' \in \mathbb{F}_2^{\ell}$. We represent $\mathbf{e}'$ as a sum of three vectors, that is, $\mathbf{e}' = \mathbf{e}_1^{(1)} + \mathbf{e}_2^{(1)} + \mathbf{e}_3^{(1)}$ with $\mathbf{e}_1^{(1)}, \mathbf{e}_2^{(1)}, \mathbf{e}_3^{(1)} \in \mathbb{F}_2^{k+\ell}$.

Since the computations are performed in the field $\mathbb{F}_2$, the 1-coordinates of $\mathbf{e}'$ can be represented in the following forms: $1+0+0, 0+1+0, 0+0+1, 1+1+1$; furthermore, the 0-coordinates can be represented as $0+1+1, 1+0+1, 1+1+0, 0+0+0$.

For arbitrary $h \geq 2$, Algorithm 4 gives the pseudocode of TER for decoding with depth $h$. For each level $i$ $(1 \leq i \leq h-1)$, we define the following quantities:

- Let $\epsilon_1^{(i)} n$ denote the number of representations of the form $1+1+1$;
- Let $\epsilon_0^{(i)} n$ denote the number of representations of the form $0+1+1$, $1+0+1$, or $1+1+0$;
- Let $p^{(i)}$ denote the number of 1-coordinates, then $p^{(0)} = p$.

**Algorithm 4** TER for decoding with depth $h$ (TER-de$^{(h)}$)

**Input:** $(P_1, P_2, \mathbf{s}', \mathbf{s}'', p) \in \mathbb{F}_2^{\ell \times (k+\ell)} \times \mathbb{F}_2^{(n-k-\ell)\times(k+\ell)} \times \mathbb{F}_2^{\ell} \times \mathbb{F}_2^{(n-k-\ell)} \times (0,1)$.

**Parameters:** $\epsilon_1^{(i)}, \epsilon_0^{(i)}$ for $1 \leq i < h$.

**Output:** $\mathbf{e}' \in \mathbb{F}_2^{k+\ell}$ of weight $p$ such that $P_1\mathbf{e}' = \mathbf{s}'$ and $|P_2\mathbf{e}' + \mathbf{s}''| = \omega - p$, or $\perp$ if no such vector is found.

```
Construct all level-h lists L_j^(h) for j ∈ {1, ..., 2 × 3^(d−1)}.
Compute L_j^(h−1) from L_(2j−1)^(h), L_(2j)^(h) for j ∈ {1, ..., 3^(h−1)}.
for i = h − 2 down to 0 do
    for j = 1 to 3^i do
        Construct L_j^(i) from L_(3j−2)^(i+1), L_(3j−1)^(i+1), L_(3j)^(i+1).
    end for
end for
if ∃e' ∈ L_1^(0) such that |P_2 e' + s''| = ω − p then
    return e'
else
    return ⊥
end if
```

The counts of these representations are summarized in Table 5. For each level $i$ $(1 \leq i \leq h-1)$, we can compute the following quantities:

- The number of representations is

$$R^{(i)} = \binom{p^{(i-1)}}{\epsilon_1^{(i)} n, \left(p^{(i-1)} - \epsilon_1^{(i)} n\right)/3, \left(p^{(i-1)} - \epsilon_1^{(i)} n\right)/3, \cdot} \times \binom{k + \ell - p^{(i-1)} - 3\epsilon_0^{(i)} n}{\epsilon_0^{(i)} n, \epsilon_0^{(i)} n, \epsilon_0^{(i)} n, \cdot};$$

- The number of 1-coordinates is

$$p^{(i)} = \frac{\left(p^{(i-1)} + 2\epsilon_1^{(i)}\right) n}{3} + \epsilon_0^{(i)} n;$$

- The list size is

$$L^{(i)} = \binom{k + \ell}{p^{(i)}} / \sqrt{R^{(i)}};$$

- The time cost is

$$T^{(i)} = \frac{\left(L^{(i)}\right)^3}{\sqrt{R^{(i-1)}/R^{(i)}}},$$

where we set $R^{(0)} = 2^{2l}$ for the consistency of the formula.

**Table 5.** Count of different level-$i$ $(1 \le i \le h-1)$ representations.

| | **Representations** | **Count** |
|---|---|---|
| 1-coordinates | $1+0+0$ | $\frac{p^{(i-1)} - \epsilon_1^{(i)} n}{3}$ |
| | $0+1+0$ | $\frac{p^{(i-1)} - \epsilon_1^{(i)} n}{3}$ |
| | $0+0+1$ | $\frac{p^{(i-1)} - \epsilon_1^{(i)} n}{3}$ |
| | $1+1+1$ | $\epsilon_1^{(i)} n$ |
| 0-coordinates | $0+1+1$ | $\epsilon_0^{(i)} n$ |
| | $1+0+1$ | $\epsilon_0^{(i)} n$ |
| | $1+1+0$ | $\epsilon_0^{(i)} n$ |
| | $0+0+0$ | $k + \ell - p^{(i-1)} - 3\epsilon_0^{(i)} n$ |

The level-$h$ list size is

$$L^{(h)} = \binom{(k+\ell)/2}{p^{(i-1)}/2},$$

and the level-$h$ runtime is

$$T^{(h)} = \frac{\left(L^{(h)}\right)^2}{\sqrt{R^{(i-1)}}}.$$

Then, the overall time complexity is

$$T(p, \ell; \epsilon_1^{(i)}, \epsilon_0^{(i)}) = \max\left(L^{(h)}, T^{(1)}, \ldots, T^{(h)}\right),$$

and the space complexity is

$$S(p, \ell; \epsilon_1^{(i)}, \epsilon_0^{(i)}) = \max\left(L^{(1)}, \ldots, L^{(h)}\right),$$

where $1 \leq i \leq h-1$.

Next, we integrate TER for decoding into the ISD framework. The complete pseudocode of the resulting ISD algorithm is provided in Algorithm 5.

**Algorithm 5** An ISD algorithm that leverages TER

**Input:** Syndrome decoding instance $(P, \mathbf{y}, \omega)$

**Parameters:** $k/n$, $p$ with $0 \leq p \leq \omega$, $\ell$ with $0 \leq \ell \leq n-k$, $h$, $\epsilon_1^{(i)}, \epsilon_0^{(i)}$ for $1 \leq i < h$.

**Output:** A solution $\mathbf{e} \in \mathbb{F}_2^n$ of weight $\omega$ such that $P\mathbf{e} = P\mathbf{y}$.

1: **repeat**

2: Randomly select a permutation matrix $\Pi \in \mathbb{F}_2^{n \times n}$.

3: Using Gaussian elimination compute $G$ such that

$$G(P\Pi^{-1}) = \begin{bmatrix} P_1 & \mathbf{0} \\ P_2 & I_{n-k-\ell} \end{bmatrix},$$

4: Compute $GPy = (\mathbf{s}', \mathbf{s}'') \in \mathbb{F}_2^{k+\ell} \times \mathbb{F}_2^{n-k-\ell}$.

5: $\mathbf{e}' \leftarrow \text{TER-de}^{(h)}(P_1, P_2, \mathbf{s}', \mathbf{s}'', p)$ ▷ Call Algorithm 4

6: **until** $\mathbf{e}' \neq \perp$

7: Output $\mathbf{e} = \Pi^{-1}(\mathbf{e}', P_2\mathbf{e}' + \mathbf{s}'')$.

To determine the computational complexity of our algorithm for a fixed code rate $k/n$ and a fixed search tree depth $h$, we aim to optimize the parameters $p, \ell$, and $\epsilon_1^{(i)}, \epsilon_0^{(i)}$ for $1 \leq i < h$, such that the expression

$$T(p, \ell; \epsilon_1^{(i)}, \epsilon_0^{(i)}) \cdot \mathcal{P}(p, \ell)^{-1}$$

is minimized, subject to the following constraints:

$$\begin{aligned} &0 < \ell < \min(n-k, n-k-\omega-p) \\ &0 < p < \min(\omega, k+\ell) \\ &0 < p^{(h-1)} < \cdots < p^{(1)} < p \\ &0 < \sqrt{R^{(h-1)}} < \cdots < \sqrt{R^{(1)}} < \ell. \end{aligned}$$

In the half-distance decoding scenario, optimization leads to complexities of

$$T_{\text{HD}} = 2^{0.0453n} \text{ and } M_{\text{HD}} = 2^{0.0293n},$$

under a worst-case code rate of $k/n = 0.444$ with $\omega/n = H^{-1}(1-k/n) = 0.065$ and the following parameter values:

$$h = 2, \frac{p}{n} = 0.015, \frac{l}{n} = 0.071,$$

$$\epsilon_0^{(1)} = 0.000000, \epsilon_1^{(1)} = 0.000004.$$

Our algorithm achieves a runtime of $T_{\text{HD}} = 2^{0.0453n}$, which represents a polynomial improvement over the previous best runtime of $2^{0.0465n}$ [41].

In the full-distance decoding scenario, optimization leads to complexities of

$$T_{\text{FD}} = 2^{0.0973n} \text{ and } M_{\text{FD}} = 2^{0.0667n},$$

under a worst-case code rate of $k/n = 0.405$ with $\omega/n = H^{-1}(1 - k/n) = 0.144$ and the following parameter values:

$$h = 3, \frac{p}{n} = 0.047, \frac{l}{n} = 0.171,$$

$$\epsilon_0^{(1)} = 0.000000, \epsilon_1^{(1)} = 0.000062,$$

$$\epsilon_0^{(2)} = 0.00000000, \epsilon_1^{(2)} = 0.00000009.$$

The running time of our algorithm, $T_{\mathrm{FD}} = 2^{0.0973n}$, is faster than the previous best result of $2^{0.1020n}$ [22] achieved using representation techniques alone. However, it does not outperform the current state-of-the-art algorithm [41], which has a time complexity of $2^{0.0885n}$ and incorporates both representation techniques and nearest neighbor search. We speculate that combining our ternary representation technique with nearest neighbor search could lead to further improvements.

### *5.3. Improved Quantum ISD Algorithm*

In this subsection, we detail the implementation of a quantum version of Algorithm 5. The quantum speedup over the classical algorithm primarily stems from two aspects: first, the accelerated search for a good permutation $\Pi$, and second, the efficient identification of a valid vector $\mathbf{e}'$ that meets the required conditions for a given permutation.

Kachigar and Tillich [23] suggested using Grover's algorithm to search for good permutations. Grover's algorithm [48] is an optimal method for unstructured search problems, offering a quadratic speedup over the best classical algorithms. The unstructured search problem involves finding an element $x$ in a set $\Sigma$ that is marked as "good" by a given oracle function $f : \Sigma \to \{0, 1\}$, where $f(x) = 1$ indicates that $x$ is good. Let $\eta$ denote the fraction of elements in $\Sigma$ for which $f(x) = 1$. Grover's algorithm finds a solution with $\mathcal{O}(1/\sqrt{\eta})$ queries to $f$—compared to a classical lower bound of $\Omega(1/\eta)$—and, assuming an average query time of $T_f$, the total execution time is $\mathcal{O}(T_f/\sqrt{\eta})$.

Algorithm 6 presents the oracle implementation for our quantum ISD algorithm. Specifically, the oracle function within the Grover search is realized through a quantum walk routine. The quantum ISD algorithm then operates by performing transformations on quantum states of the following form (normalization omitted):

$$\sum_{i=1}^{\mathcal{P}(p,\ell)} |\Pi_i\rangle|\Pi_i(P)\rangle \otimes \underbrace{\left[\sum_{U_j^{(h)} \in J_j(N,k)} |U_1^{(h)}\rangle \otimes \cdots \otimes |U_{2\times 3^{h-1}}^{(h)}\rangle \otimes |\mathrm{aux}\rangle \otimes |U_1^{(0)}\rangle\right]}_{\text{Quantum Walk = Check for the outer Grover}} \otimes |\text{Is } \Pi \text{ good?}\rangle$$

The outer search applies Grover's algorithm over a set of $\mathcal{P}(p, \ell)$ permutations. The validity of a permutation $\Pi$ is verified via a quantum walk search, as detailed in Algorithm 6. After approximately $T_{\mathrm{QW}}$ steps, the register $|U_1^{(0)}\rangle$ contains a non-empty level-0 list that yields the correct error vector provided that $\Pi$ is a good permutation.

Thus, the total complexity of our quantum ISD algorithm is $\mathcal{O}(\sqrt{\mathcal{P}(p,\ell)} \cdot T_{\mathrm{QW}})$. Optimization leads to complexities of

$$T_{\mathrm{QISD}} = 2^{0.058326n} \text{ and } M_{\mathrm{QISD}} = 2^{0.020802n},$$

under a worst-case code rate of $k/n = 0.447$ with $\omega/n = H^{-1}(1 - k/n) = 0.128$ and the following parameter values:

$$h = 2, \frac{p}{n} = 0.013, \frac{l}{n} = 0.052,$$

$$\epsilon_0^{(1)} = 0.000000, \epsilon_1^{(1)} = 0.000011.$$

Our quantum ISD algorithm achieves a runtime of $T_{\text{QISD}} = 2^{0.058326n}$, yielding a polynomial improvement over the previous best quantum runtime of $2^{0.058696n}$ [23].

**Algorithm 6** An oracle for quantum ISD algorithm

1: [Setup] Prepare the initial state (normalization omitted):

$$\left[\sum_{U_j^{(h)} \in J_j(N,k)} \bigotimes_{j=1} |U_j^{(h)}\rangle \, |\text{aux}\rangle \, |0\rangle\right] |0\rangle,$$

where $h$ denotes the depth of the search tree, the registers within the brackets are employed in the quantum walk search, and the last register is used to store the output of the oracle.

2: **Repeat** $\mathcal{O}(1/\sqrt{\epsilon})$ times:

(2.1) [Checking] Perform the quantum walk checking.

(2.2) [Update] **Repeat** $\mathcal{O}(1/\sqrt{\delta})$ times:

(2.2.1) Perform a single step of the quantum walk.

(2.2.2) Update the data structure.

3: The quantum state at this point is

$$\left[\sum_{U_j^{(h)} \in J_j(N,k)} \bigotimes_{j=1} |U_j^{(h)}\rangle \, |\text{aux}\rangle \, |U_1^{(0)}\rangle\right] |0\rangle.$$

A controlled operation is then applied, using the register storing $U_1^{(0)}$ as the control and the last register as the target, such that the bit in the last register is flipped if $U_1^{(0)}$ is non-empty.

## 6. Conclusions

In this work, we have presented significant advancements in solving RSSP and, by extension, the decoding problem for random linear codes in both the classical and quantum computational settings.

Our primary contribution is the introduction of a novel heuristic algorithm for RSSP based on a ternary tree representation structure. This approach, inspired by techniques in lattice-based cryptanalysis, fundamentally departs from the binary tree structures employed by all previous classical heuristic algorithms. Through meticulous numerical optimization, we demonstrated that our classical algorithm achieves a time complexity of $\widetilde{\mathcal{O}}\left(2^{0.2400n}\right)$ and a space complexity of $\widetilde{\mathcal{O}}\left(2^{0.2221n}\right)$, establishing a new state of the art for classical heuristic algorithms.

Building upon this classical foundation, we developed a novel quantum algorithm by integrating the ternary representation technique with a quantum walk search framework. The resulting algorithm achieves a time and space complexity of $\widetilde{\mathcal{O}}\left(2^{0.1843n}\right)$, which, to the best of our knowledge, represents the best-known heuristic performance for RSSP on quantum computers.

We further demonstrated the cryptographic impact of our algorithms by applying them to the decoding problem. For half-distance decoding, our classical algorithm improves upon the state-of-the-art BM algorithm, reducing the time complexity from $\widetilde{\mathcal{O}}(2^{0.0465n})$ to $\widetilde{\mathcal{O}}(2^{0.0453n})$. For full-distance decoding, while our result of $\widetilde{\mathcal{O}}(2^{0.0973n})$ improves upon pure representation-based approaches like BJMM, it does not surpass the hybrid technique of

Both–May. However, this strongly suggests a promising avenue for future work: integrating our ternary representation techniques with nearest neighbor search techniques could potentially lead to further breakthroughs in decoding complexity. In the quantum setting, our proposed ISD algorithm outperforms all previous works, achieving a time complexity of $\widetilde{\mathcal{O}}(2^{0.058326n})$.

Finally, while the quantum algorithms presented in this work achieve improved asymptotic complexities, their practical implementation remains challenging due to the exponential number of qubits required. Recent research directions have explored alternative quantum approaches that utilize only polynomial qubit resources, albeit at the cost of exponential classical memory or higher quantum time complexity [49,50]. Investigating such space-efficient implementations, which may be more amenable to future quantum hardware, represents an interesting direction for future work.

**Funding:** This work was supported by the Innovation Program for Quantum Science and Technology under Grant No. 2024ZD0300502 and the Beijing Nova Program under Grants No. 20220484128 and 20240484652.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The code associated with this submission is publicly available and can be accessed via the following link: https://gitee.com/li-yang777777/subset-sum (accessed on 22 August 2025).

**Acknowledgments:** The authors thank the reviewers for their valuable feedback and insightful suggestions, which greatly improved this manuscript.

**Conflicts of Interest:** The author declares no conflicts of interest.

## Appendix A. TER with Arbitrary Depth

For arbitrary $d \geq 2$, Algorithm A1 gives the pseudocode of TER with depth $d$. Following the same analytical approach as in Section 3, we start from the first level and compute the number of representations and expected list sizes for each level up to level $d$.

For each level $i$ ($1 \leq i \leq d-1$), we define the following quantities:

- Let $\epsilon_1^{(i)} n$ denote the number of representations of the form $1+1+(-1)$, $1+(-1)+1$, or $(-1)+1+1$.
- Let $\epsilon_0^{(i)} n$ denote the number of representations of the form $0+1+(-1)$, $0+(-1)+1$, $1+0+(-1)$, $(-1)+0+1$, $1+(-1)+0$, or $(-1)+1+0$.
- Let $\epsilon_{-1}^{(i)} n$ denote the number of representations of the form $1+(-1)+(-1)$, $(-1)+1+(-1)$, or $(-1)+(-1)+1$.
- Let $w^{(i)} n$ denote the number of $-1$-coordinates.
- Let $v^{(i)} n$ denote the number of 1-coordinates.

The result from level 1 is the same as the depth-3 case. Table 3 presents the number of each type of level-1 representations. Thus, $w^{(1)} = \epsilon_1^{(1)} + 2\epsilon_0^{(1)}$, $v^{(1)} = 1/6 + w^{(1)}$, and the number of representations is

$$R^{(1)} = \binom{n/2}{\epsilon_1^{(1)} n, \epsilon_1^{(1)} n, \epsilon_1^{(1)} n, (1/6 - \epsilon_1^{(1)}) n, (1/6 - \epsilon_1^{(1)}) n, \cdot} \times \binom{n/2}{\epsilon_0^{(1)} n, \epsilon_0^{(1)} n, \epsilon_0^{(1)} n, \epsilon_0^{(1)} n, \epsilon_0^{(1)} n, \epsilon_0^{(1)} n, \cdot}.$$

Table A1 presents the number of each type of level-$i$ ($2 \le i \le d-1$) representation. Then, for $2 \le i \le d-1$,

$$w^{(i)} = w^{(i-1)}/3 + \epsilon_{-1}^{(i)} + \epsilon_1^{(i)} + 2\epsilon_0^{(i)},$$
$$v^{(i)} = \frac{1}{2 \times 3^i} + w^{(i)}.$$

and the number of representations can be computed as follows:

$$R^{(i)} = \binom{v^{(i-1)}n}{\epsilon_1^{(i)}n, \epsilon_1^{(i)}n, \epsilon_1^{(i)}n, \left(\frac{v^{(i-1)}}{3} - \epsilon_1^{(i)}\right)n, \left(\frac{v^{(i-1)}}{3} - \epsilon_1^{(i)}\right)n, \cdot} \times \binom{\left(1 - v^{(i-1)} - w^{(i-1)}\right)n}{\epsilon_0^{(i)}n, \epsilon_0^{(i)}n, \epsilon_0^{(i)}n, \epsilon_0^{(i)}n, \epsilon_0^{(i)}n, \epsilon_0^{(i)}n, \cdot} \times \binom{w^{(i-1)}n}{\epsilon_{-1}^{(i)}n, \epsilon_{-1}^{(i)}n, \epsilon_{-1}^{(i)}n, \left(\frac{w^{(i-1)}}{3} - \epsilon_{-1}^{(i)}\right)n, \left(\frac{w^{(i-1)}}{3} - \epsilon_{-1}^{(i)}\right)n, \cdot}.$$

Let $L^{(i)}$ be the size of lists at level $i$ for $1 \le i < d$. Then, under Heuristic 2

$$L^{(i)} = \binom{n}{w^{(i)}n, v^{(i)}n, \cdot} / \sqrt{R^{(i)}},$$
$$L^{(d)} = \binom{n/2}{(w^{(d-1)}/2)n, (v^{(d-1)}/2)n, \cdot}.$$

The time complexity of Algorithm A1 is

$$\max\left(L^{(d)}, \frac{\left(L^{(d)}\right)^2}{\sqrt{R^{(d-1)}}}, \frac{\left(L^{(d-2)}\right)^3}{\sqrt{R^{(d-2)}/R^{(d-1)}}}, \cdots, \frac{\left(L^{(1)}\right)^3}{N/\sqrt{R^{(1)}}}\right),$$

and the space complexity is $\max\left(L^{(d)}, \cdots, L^{(1)}\right)$.

**Algorithm A1** TER with depth $d$

**Input:** The random subset-sum instance $(\mathbf{a}, s) \in (\mathbb{Z}_N)^{n+1}$

**Parameters:** $\epsilon_1^{(1)}, \epsilon_0^{(1)}, \epsilon_1^{(i)}, \epsilon_0^{(i)}, \epsilon_{-1}^{(i)}$ for $2 \le i < d$.

**Output:** A solution $\mathbf{e} \in D^n\left[0, 1/2\right]$ or $\perp$ if no solution is found.

Construct all level-$d$ lists $L_j^{(d)}$ for $j \in \{1, \ldots, 2 \times 3^{d-1}\}$.
Compute $L_j^{(d-1)}$ from $L_{2j-1}^{(d)}, L_{2j}^{(d)}$ for $j \in \{1, \ldots, 3^{d-1}\}$.
**for** $i = d-2$ down to 0 **do**
    **for** $j = 1$ to $3^i$ **do**
        Construct $L_j^{(i)}$ from $L_{3j-2}^{(i+1)}, L_{3j-1}^{(i+1)}, L_{3j}^{(i+1)}$.
    **end for**
**end for**
**if** $L_1^{(0)} \neq \emptyset$ **then**
    **return** any $\mathbf{e} \in L_1^{(0)}$
**else**
    **return** $\perp$
**end if**

**Table A1.** Count of different level-$i$ $(2 \leq i \leq d-1)$ representations.

| | Representations | Count |
|---|---|---|
| 1-coordinates | $1+0+0$ | $\left(\frac{v^{(i-1)}}{3}-\epsilon_1^{(i)}\right)n$ |
| | $0+1+0$ | $\left(\frac{v^{(i-1)}}{3}-\epsilon_1^{(i)}\right)n$ |
| | $0+0+1$ | $\left(\frac{v^{(i-1)}}{3}-\epsilon_1^{(i)}\right)n$ |
| | $1+1+(-1)$ | $\epsilon_1^{(i)}n$ |
| | $1+(-1)+1$ | $\epsilon_1^{(i)}n$ |
| | $(-1)+1+1$ | $\epsilon_1^{(i)}n$ |
| 0-coordinates | $0+1+(-1)$ | $\epsilon_0^{(i)}n$ |
| | $0+(-1)+1$ | $\epsilon_0^{(i)}n$ |
| | $1+0+(-1)$ | $\epsilon_0^{(i)}n$ |
| | $(-1)+0+1$ | $\epsilon_0^{(i)}n$ |
| | $1+(-1)+0$ | $\epsilon_0^{(i)}n$ |
| | $(-1)+1+0$ | $\epsilon_0^{(i)}n$ |
| | $0+0+0$ | $\left(1-v^{(i-1)}-w^{(i-1)}-6\epsilon_0^{(i)}\right)n$ |
| $-1$-coordinates | $(-1)+0+0$ | $\left(\frac{w^{(i-1)}}{3}-\epsilon_{-1}^{(i)}\right)n$ |
| | $0+(-1)+0$ | $\left(\frac{w^{(i-1)}}{3}-\epsilon_{-1}^{(i)}\right)n$ |
| | $0+0+(-1)$ | $\left(\frac{w^{(i-1)}}{3}-\epsilon_{-1}^{(i)}\right)n$ |
| | $1+(-1)+(-1)$ | $\epsilon_{-1}^{(i)}n$ |
| | $(-1)+1+(-1)$ | $\epsilon_{-1}^{(i)}n$ |
| | $(-1)+(-1)+1$ | $\epsilon_{-1}^{(i)}n$ |

Table A2 presents the time and space complexities of algorithms TER and QTER at various depths.

**Table A2.** Time and space complexities of TER and QTER at different depths.

| Algorithm | Depth | Time Complexity | Space Complexity |
|---|---|---|---|
| TER | 2 | $\widetilde{\mathcal{O}}(2^{0.3251n})$ | $\widetilde{\mathcal{O}}(2^{0.3251n})$ |
| | 3 | $\widetilde{\mathcal{O}}(2^{0.2400n})$ | $\widetilde{\mathcal{O}}(2^{0.2221n})$ |
| | 4 | $\widetilde{\mathcal{O}}(2^{0.2400n})$ | $\widetilde{\mathcal{O}}(2^{0.2221n})$ |
| | 5 | $\widetilde{\mathcal{O}}(2^{0.2400n})$ | $\widetilde{\mathcal{O}}(2^{0.2221n})$ |
| | 6 | $\widetilde{\mathcal{O}}(2^{0.3017n})$ | $\widetilde{\mathcal{O}}(2^{0.2164n})$ |
| QTER | 2 | $\widetilde{\mathcal{O}}(2^{0.2786n})$ | $\widetilde{\mathcal{O}}(2^{0.2786n})$ |
| | 3 | $\widetilde{\mathcal{O}}(2^{0.1843n})$ | $\widetilde{\mathcal{O}}(2^{0.1843n})$ |
| | 4 | $\widetilde{\mathcal{O}}(2^{0.2289n})$ | $\widetilde{\mathcal{O}}(2^{0.2014n})$ |

## Appendix B. Filtering Probability

In this section, we provide the formula for calculating the filtering probability, which is used in the numerical optimization of QTER.

**Lemma A1.** *For $0 \le i \le d-2$, let $\mathbf{e}_1^{(i+1)}, \mathbf{e}_2^{(i+1)}, \mathbf{e}_3^{(i+1)} \in D^n\left[w^{(i+1)}, 1/(2\times 3^{i+1})\right]$. Then the level-i filtering probability of TER with depth d, denoted $pr^{(i)}$, is given by*

$$\begin{aligned}
&\binom{v^{(i+1)}n}{\epsilon_1^{(i+1)}n, \epsilon_1^{(i+1)}n, \epsilon_0^{(i+1)}n, \epsilon_0^{(i+1)}n, \epsilon_{-1}^{(i+1)}n, \cdot} \\
&\times \binom{w^{(i+1)}n}{\epsilon_1^{(i+1)}n, \epsilon_0^{(i+1)}n, \epsilon_0^{(i+1)}n, \epsilon_{-1}^{(i+1)}n, \epsilon_{-1}^{(i+1)}n, \cdot} \\
&\times \binom{\left(1-v^{(i+1)}-w^{(i+1)}\right)n}{\epsilon_0^{(i+1)}n, \epsilon_0^{(i+1)}n, \left(\frac{v^{(i)}}{3}-\epsilon_1^{(i+1)}n\right)n, \left(\frac{v^{(i)}}{3}-\epsilon_1^{(i+1)}n\right)n, \left(\frac{w^{(i)}}{3}-\epsilon_{-1}^{(i+1)}n\right)n, \left(\frac{w^{(i)}}{3}-\epsilon_{-1}^{(i+1)}n\right)n, \cdot} \\
&\div \binom{n}{v^{(i+1)}n, w^{(i+1)}n, \cdot}^2.
\end{aligned} \tag{A1}$$

**Proof.** Given a fixed vector $\mathbf{e}_1^{(i+1)} \in D^n\left[w^{(i+1)}, 1/(2\times 3^{i+1})\right]$, we proceed to count the number of pairs $\mathbf{e}_2^{(i+1)}, \mathbf{e}_3^{(i+1)} \in D^n\left[w^{(i+1)}, 1/(2\times 3^{i+1})\right]$ that are compatible with $\mathbf{e}_1^{(i+1)}$.

We begin by considering the coordinates where $\mathbf{e}_1^{(i+1)}$ has a value of 1. The total number of such coordinates is $v^{(i+1)}n = \left(w^{(i+1)} + \frac{1}{2\times 3^{i+1}}\right)n$. According to Table A1, these coordinates are categorized into the following six types based on the values that $\mathbf{e}_2^{(i+1)}$ and $\mathbf{e}_3^{(i+1)}$ take at the same positions:

- Type $1+0+0$, which occurs $v^{(i)}/3 - \epsilon_1^{(i+1)}n$ times.
- Type $1+1+(-1)$, which occurs $\epsilon_1^{(i+1)}n$ times.
- Type $1+(-1)+1$, which occurs $\epsilon_1^{(i+1)}n$ times.
- Type $1+0+(-1)$, which occurs $\epsilon_0^{(i+1)}n$ times.
- Type $1+(-1)+0$, which occurs $\epsilon_0^{(i+1)}n$ times.
- Type $1+(-1)+(-1)$, which occurs $\epsilon_{-1}^{(i+1)}n$ times.

The number of ways to assign the values of $\mathbf{e}_2^{(i+1)}$ and $\mathbf{e}_3^{(i+1)}$ at these $v^{(i+1)}n$ positions, according to the above distribution, is given by the multinomial coefficient:

$$\binom{v^{(i+1)}n}{\epsilon_1^{(i+1)}n, \epsilon_1^{(i+1)}n, \epsilon_0^{(i+1)}n, \epsilon_0^{(i+1)}n, \epsilon_{-1}^{(i+1)}n, \cdot}.$$

This corresponds to the first term in Expression (A1).

Similarly, the number of compatible pairs for the coordinates where $\mathbf{e}_1^{(i+1)}$ takes the value 0 is given by the second term in Expression (A1), and the number for the coordinates where $\mathbf{e}_1^{(i+1)}$ equals $-1$ corresponds to the third term in Expression (A1).

In order to obtain the probability, we divide the first three terms in Expression (A1) by

$$\binom{n}{v^{(i+1)}n, w^{(i+1)}n, \cdot}^2,$$

which corresponds to the total number of all possible pairs $(\mathbf{e}_2^{(i+1)}, \mathbf{e}_3^{(i+1)})$. □

## Appendix C. Proof of Theorem 3

We proceed to prove the result stated in Theorem 3 of the main text.

**Proof.** Following [46], employ exponential sums with $e(z) = \exp(2\pi i z/M)$ and the identity

$$\sum_{\lambda=0}^{M-1} e(\lambda u) = \begin{cases} 0, & \text{if } u \not\equiv 0 \pmod{M}; \\ M, & \text{if } u \equiv 0 \pmod{M}. \end{cases} \tag{A2}$$

First prove that for $\lambda_1, \lambda_2 \neq 0$ the following identity holds:

$$\sum_{\mathbf{a}\in\mathbb{Z}_M^n}\left|\sum_{\mathbf{x}\in\mathcal{B}_1}\sum_{\mathbf{y}\in\mathcal{B}_2} e(\lambda_1\mathbf{a}\cdot\mathbf{x}+\lambda_2\mathbf{a}\cdot\mathbf{y})\right|^2 = M^n|\mathcal{B}_1||\mathcal{B}_2|. \tag{A3}$$

Calculate the left side of the above identity

$$\sum_{\mathbf{a}\in\mathbb{Z}_M^n}\left|\sum_{\mathbf{x}\in\mathcal{B}_1}\sum_{\mathbf{y}\in\mathcal{B}_2} e(\lambda_1\mathbf{a}\cdot\mathbf{x}+\lambda_2\mathbf{a}\cdot\mathbf{y})\right|^2$$
$$= \sum_{\mathbf{x},\mathbf{x}'\in\mathcal{B}_1}\sum_{\mathbf{y},\mathbf{y}'\in\mathcal{B}_2}\sum_{\mathbf{a}\in\mathbb{Z}_M^n} e\left[\lambda_1(\mathbf{a}\cdot\mathbf{x}-\mathbf{a}\cdot\mathbf{x}')+\lambda_2(\mathbf{a}\cdot\mathbf{y}-\mathbf{a}\cdot\mathbf{y}')\right]$$

By Equation (A2), if $\mathbf{x}=\mathbf{x}'$ and $\mathbf{y}=\mathbf{y}'$, the inner sum contributes $M^n$. If $\mathbf{x}\neq\mathbf{x}'$ or $\mathbf{y}\neq\mathbf{y}'$, assuming $y_n \neq y_n'$ without loss of generality:

$$\sum_{\mathbf{a}\in\mathbb{Z}_M^n} e\left[\lambda_1(\mathbf{a}\cdot\mathbf{x}-\mathbf{a}\cdot\mathbf{x}')+\lambda_2(\mathbf{a}\cdot\mathbf{y}-\mathbf{a}\cdot\mathbf{y}')\right]$$
$$= \sum_{\mathbf{a}\in\mathbb{Z}_M^n} e\left[\lambda_1(\mathbf{a}\cdot\mathbf{x}-\mathbf{a}\cdot\mathbf{x}')\right] \sum_{(a_1,\cdots,a_{n-1})\in\mathbb{Z}_M^{n-1}} e\left[\lambda_2\sum_{j=1}^{n-1} a_j(y_j-y_j')\right]\sum_{a_n=0}^{M-1} e\left[\lambda_2 a_n(y_n-y_n')\right] = 0,$$

establishing Equation (A3).

For fixed $\mathbf{a}$, independence of $\mathbf{a}\cdot\mathbf{x}$ and $\mathbf{a}\cdot\mathbf{y}$ gives

$$P_{\mathbf{a}}(\mathcal{B}_1\times\mathcal{B}_2, c_1, c_2) = P_{\mathbf{a}}(\mathcal{B}_1, c_1)\cdot P_{\mathbf{a}}(\mathcal{B}_2, c_2).$$

From Equation (A2),

$$P_{\mathbf{a}}(\mathcal{B}_1, c_1) = \frac{1}{M|\mathcal{B}_1|}\sum_{\mathbf{x}\in\mathcal{B}_1}\sum_{\lambda_1=0}^{M-1} e[\lambda_1(\mathbf{a}\cdot\mathbf{x}-c_1)],$$

$$P_{\mathbf{a}}(\mathcal{B}_2, c_2) = \frac{1}{M|\mathcal{B}_2|}\sum_{\mathbf{y}\in\mathcal{B}_2}\sum_{\lambda_2=0}^{M-1} e[\lambda_2(\mathbf{a}\cdot\mathbf{y}-c_2)].$$

Thus,

$$P_{\mathbf{a}}(\mathcal{B}_1\times\mathcal{B}_2, c_1, c_2) = \frac{1}{M^2|\mathcal{B}_1||\mathcal{B}_2|}\sum_{\mathbf{x}\in\mathcal{B}_1}\sum_{\mathbf{y}\in\mathcal{B}_2}\sum_{\lambda_1,\lambda_2=0}^{M-1} e[\lambda_1(\mathbf{a}\cdot\mathbf{x}-c_1)+\lambda_2(\mathbf{a}\cdot\mathbf{y}-c_2)].$$

The contribution of the term corresponding to $\lambda_1=\lambda_2=0$ is $\frac{1}{M^2}$.
Therefore,

$$\sum_{c_1,c_2\in\mathbb{Z}_M}\left[P_{\mathbf{a}}(\mathcal{B}_1\times\mathcal{B}_2, c_1, c_2)-\frac{1}{M^2}\right]^2$$
$$= \sum_{c_1,c_2\in\mathbb{Z}_M}\left[\frac{1}{M^2|\mathcal{B}_1||\mathcal{B}_2|}\sum_{\lambda_1,\lambda_2=1}^{M-1} e(-\lambda_1c_1-\lambda_2c_2)\sum_{\mathbf{x}\in\mathcal{B}_1}\sum_{\mathbf{y}\in\mathcal{B}_2} e(\lambda_1\mathbf{a}\cdot\mathbf{x}+\lambda_2\mathbf{a}\cdot\mathbf{y})\right]^2$$
$$= \frac{1}{M^4|\mathcal{B}_1|^2|\mathcal{B}_2|^2}\sum_{c_1,c_2\in\mathbb{Z}_M}\sum_{\lambda_1,\lambda_1',\lambda_2,\lambda_2'=1}^{M-1} e\left[-c_1(\lambda_1+\lambda_1')-c_2(\lambda_2+\lambda_2')\right]$$
$$\sum_{\mathbf{x},\mathbf{x}'\in\mathcal{B}_1}\sum_{\mathbf{y},\mathbf{y}'\in\mathcal{B}_2} e(\lambda_1\mathbf{a}\cdot\mathbf{x}+\lambda_1'\mathbf{a}\cdot\mathbf{x}'+\lambda_2\mathbf{a}\cdot\mathbf{y}+\lambda_2'\mathbf{a}\cdot\mathbf{y}')$$

$$= \frac{1}{M^4|\mathcal{B}_1|^2|\mathcal{B}_2|^2} \sum_{\lambda_1,\lambda_1',\lambda_2,\lambda_2'=1}^{M-1} \sum_{\mathbf{x},\mathbf{x}'\in\mathcal{B}_1} \sum_{\mathbf{y},\mathbf{y}'\in\mathcal{B}_2} e(\lambda_1\mathbf{a}\cdot\mathbf{x} + \lambda_1'\mathbf{a}\cdot\mathbf{x}' + \lambda_2\mathbf{a}\cdot\mathbf{y} + \lambda_2'\mathbf{a}\cdot\mathbf{y}')$$
$$\sum_{c_1,c_2\in\mathbb{Z}_M} e\big[-c_1(\lambda_1+\lambda_1') - c_2(\lambda_2+\lambda_2')\big].$$

The sum over $c_1, c_2$ vanishes if $\lambda_1 + \lambda_1' \not\equiv 0 \pmod{M}$ or $\lambda_2 + \lambda_2' \not\equiv 0 \pmod{M}$ and is equal to $M$ otherwise. Hence,

$$\sum_{c_1,c_2\in\mathbb{Z}_M} \left[P_{\mathbf{a}}(\mathcal{B}_1\times\mathcal{B}_2, c_1, c_2) - \frac{1}{M^2}\right]^2$$
$$= \frac{1}{M^2|\mathcal{B}_1|^2|\mathcal{B}_2|^2} \sum_{\lambda_1,\lambda_2=1}^{M-1} \sum_{\mathbf{x},\mathbf{x}'\in\mathcal{B}_1} \sum_{\mathbf{y},\mathbf{y}'\in\mathcal{B}_2} e\big[\lambda_1(\mathbf{a}\cdot\mathbf{x} - \mathbf{a}\cdot\mathbf{x}') + \lambda_2(\mathbf{a}\cdot\mathbf{y} + \mathbf{a}\cdot\mathbf{y}')\big]$$
$$= \frac{1}{M^2|\mathcal{B}_1|^2|\mathcal{B}_2|^2} \sum_{\lambda_1,\lambda_2=1}^{M-1} \left|\sum_{\mathbf{x}\in\mathcal{B}_1} \sum_{\mathbf{y}\in\mathcal{B}_2} e(\lambda_1\mathbf{a}\cdot\mathbf{x} + \lambda_2\mathbf{a}\cdot\mathbf{y})\right|^2$$

Applying Equation (A3) yields the result. □

## References


1. Garey, M.R.; Johnson, D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*; W.H. Freeman: San Francisco, CA, USA, 1979; p. 338.
2. Merkle, R.; Hellman, M. Hiding information and signatures in trapdoor knapsacks. *IEEE Trans. Inf. Theory* **1978**, *24*, 525–530. [CrossRef]
3. Chor, B.; Rivest, R.L. A knapsack-type public key cryptosystem based on arithmetic in finite fields. *IEEE Trans. Inf. Theory* **1988**, *34*, 901–909. [CrossRef]
4. Impagliazzo, R.; Naor, M. Efficient cryptographic schemes provably as secure as subset sum. *J. Cryptol.* **1996**, *9*, 199–216. [CrossRef]
5. Faust, S.; Masny, D.; Venturi, D. Chosen-ciphertext security from subset sum. In *Public-Key Cryptography—PKC 2016*; Springer International Publishing: Berlin/Heidelberg, Germany, 2016; pp. 35–46.
6. Lyubashevsky, V.; Palacio, A.; Segev, G. Public-key cryptographic primitives provably as secure as subset sum. In Proceedings of the Theory of Cryptography Conference, Zurich, Switzerland, 9–11 February 2010; Springer International Publishing: Berlin/Heidelberg, Germany, 2010; pp. 382–400.
7. Bonnetain, X. Improved low-qubit hidden shift algorithms. *arXiv* **2019**, arXiv:1901.11428. [CrossRef]
8. Bonnetain, X.; Schrottenloher, A. Quantum security analysis of CSIDH. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, 10–14 May 2020; Springer International Publishing: Berlin/Heidelberg, Germany, 2020; pp. 493–522.
9. Bonnetain, X.; Naya-Plasencia, M. Hidden shift quantum cryptanalysis and implications. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, 2–6 December 2018; Springer International Publishing: Berlin/Heidelberg, Germany, 2018; pp. 560–592.
10. Galil, Z.; Margalit, O. An almost linear-time algorithm for the dense subset-sum problem. *SIAM J. Comput.* **1991**, *20*, 1157–1189. [CrossRef]
11. Brickell, E.F. Solving low density knapsacks. In *Advances in Cryptology: Proceedings of Crypto 83*; Springer US: Boston, MA, USA, 1984; pp. 25–37.
12. Lagarias, J.C.; Odlyzko, A.M. Solving low-density subset sum problems. *J. ACM (JACM)* **1985**, *32*, 229–246. [CrossRef]
13. Ajtai, M. The shortest vector problem in $L_2$ is np-hard for randomized reductions. In Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, Dallas, TX, USA, 24–26 May 1998; Association for Computing Machinery: New York, NY, USA, 1998; pp. 10–19.
14. Coster, M.J.; LaMacchia, B.A.; Odlyzko, A.M.; Schnorr, C.P. An improved low-density subset sum algorithm. In Proceedings of the Workshop on the Theory and Application of of Cryptographic Techniques, Brighton, UK, 8–11 April 1991; Springer International Publishing: Berlin/Heidelberg, Germany, 1991; pp. 54–67.
15. Joux, A.; Stern, J. Improving the critical density of the Lagarias-Odlyzko attack against subset sum problems. In Proceedings of the International Symposium on Fundamentals of Computation Theory, Gosen, Germany, 9–13 September 1991; Springer International Publishing: Berlin/Heidelberg, Germany, 1991; pp. 258–264.

16. Horowitz, E.; Sahni, S. Computing partitions with applications to the knapsack problem. *J. ACM (JACM)* **1974**, *21*, 277–292. [CrossRef]
17. Schroeppel, R.; Shamir, A. A $t = O(2^{n/2})$, $s = O(2^{n/4})$ algorithm for certain np-complete problems. *SIAM J. Comput.* **1981**, *10*, 456–464. [CrossRef]
18. Howgrave-Graham, N.; Joux, A. New generic algorithms for hard knapsacks. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, 10–14 May 2020; Springer International Publishing: Berlin/Heidelberg, Germany, 2010; pp. 235–256.
19. Becker, A.; Coron, J.; Joux, A. Improved generic algorithms for hard knapsacks. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, 10–14 May 2020; Springer International Publishing: Berlin/Heidelberg, Germany, 2011; pp. 364–385.
20. Bonnetain, X.; Bricout, R.; Schrottenloher, A.; Shen, Y. Improved classical and quantum algorithms for subset-sum. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, 2–6 December 2018; Springer International Publishing: Berlin/Heidelberg, Germany, 2020; pp. 633–666.
21. May, A.; Meurer, A.; Thomae, E. Decoding random linear codes in $\widetilde{\mathcal{O}}(2^{0.054n})$. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, 2–6 December 2018; Springer International Publishing: Berlin/Heidelberg, Germany, 2011; pp. 107–124.
22. Becker, A.; Joux, A.; May, A.; Meurer, A. Decoding random binary linear codes in $2^{n/20}$: How $1 + 1 = 0$ improves information set decoding. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, 10–14 May 2020; Springer International Publishing: Berlin/Heidelberg, Germany, 2012; pp. 520–536.
23. Kachigar, G.; Tillich, J. Quantum information set decoding algorithms. In Proceedings of the International Workshop on Post-Quantum Cryptography, Utrecht, The Netherlands, 26–28 June 2017; Springer International Publishing: Cham, Switzerland, 2017; pp. 69–89.
24. Kirshanova, E. Improved quantum information set decoding. In Proceedings of the International Workshop on Post-Quantum Cryptography, Utrecht, The Netherlands, 26–28 June 2017; Springer International Publishing: Cham, Switzerland, 2018; pp. 507–527.
25. May, A. How to meet ternary LWE keys. In Proceedings of the Annual International Cryptology Conference, Virtual, 16–20 August 2021; Springer International Publishing: Cham, Switzerland, 2021; pp. 701–731.
26. van Hoof, I.; Kirshanova, E.; May, A. Quantum Key Search for Ternary LWE. In Proceedings of the International Workshop on Post-Quantum Cryptography, Utrecht, The Netherlands, 26–28 June 2017; Springer International Publishing: Cham, Switzerland, 2021; pp. 117–132.
27. Kirshanova, E.; May, A. How to Find Ternary LWE Keys Using Locality Sensitive Hashing. In Proceedings of the IMA International Conference on Cryptography and Coding, Oxford, UK, 14–16 December 2021.
28. Glaser, T.; May, A. How to Enumerate LWE Keys as Narrow as in Kyber/Dilithium. In Proceedings of the International Conference on Cryptology and Network Security, Singapore, 19–22 December 2023.
29. Lee, E.; Lee, J.; Son, Y.; Wang, Y. Improved Meet-LWE Attack via Ternary Trees. Cryptology ePrint Archive, Paper 2024/824. Available online: https://eprint.iacr.org/2024/824 (accessed on 23 August 2025).
30. Bernstein, D.J.; Jeffery, S.; Lange, T.; Meurer, A. Quantum algorithms for the subset-sum problem. In Proceedings of the International Workshop on Post-Quantum Cryptography, Utrecht, The Netherlands, 26–28 June 2017; Springer International Publishing: Berlin/Heidelberg, Germany, 2013; pp. 16–33.
31. Helm, A.; May, A. Subset Sum Quantumly in $1.17^n$. In Proceedings of the 13th Conference on the Theory of Quantum Computation, Communication and Cryptography, Sydney, Australia, 16–18 July 2018; Schloss Dagstuhl–Leibniz-Zentrum für Informatik: Dagstuhl, Germany, 2018; pp. 5:1–5:15.
32. Magniez, F.; Nayak, A.; Roland, J.; Santha, M. Search via quantum walk. *SIAM J. Comput.* **2011**, *40*, 142–164. [CrossRef]
33. NIST. Status Report on the Fourth Round of the NIST Post-Quantum Cryptography Standardization Process. NIST Internal Report 8545. Available online: https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization (accessed on 3 March 2025).
34. Albrecht, M.R.; Bernstein, D.J.; Chou, T.; Cid, C.; Gilcher, J.; Lange, T.; Maram, V.; Von Maurich, I.; Misoczki, R.; Niederhagen, R.; et al. Classic McEliece: Conservative Code-Based Cryptography. In *NIST Post-Quantum Standardization, 4th Round*; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2022. Available online: https://classic.mceliece.org/nist.html (accessed on 23 October 2022).
35. Aragon, N.; Barreto, P.; Bettaieb, S.; Bidoux, L.; Blazy, O.; Deneuville, J.C.; Gaborit, P.; Ghosh, S.; Gueron, S.; Güneysu, T.; et al. BIKE: Bit Flipping Key Encapsulation. In *NIST Post-Quantum Standardization, 4th Round*; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2024. Available online: https://bikesuite.org/ (accessed on 10 October 2024).

36. Gaborit, P.; Aguilar-Melchor, C.; Aragon, N.; Bettaieb, S.; Bidoux, L.; Blazy, O.; Deneuville, J.C.; Persichetti, E.; Zémor, G.; Bos, J.; et al. Hamming Quasi-Cyclic (HQC). In *NIST Post-Quantum Standardization, 4th Round*; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2025. Available online: https://pqc-hqc.org/resources.html (accessed on 22 August 2025).
37. Prange, E. The use of information sets in decoding cyclic codes. *IRE Trans. Inf. Theory* **1962**, *8*, 5–9. [CrossRef]
38. Stern, J. A method for finding codewords of small weight. In Proceedings of the International Colloquium on Coding Theory and Applications, Toulon, France, 2–4 November 1988; Springer International Publishing: Berlin/Heidelberg, Germany, 1988; pp. 106–113.
39. Finiasz, M.; Sendrier, N. Security bounds for the design of code-based cryptosystems. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, 2–6 December 2018; Springer International Publishing: Berlin/Heidelberg, Germany, 2009; pp. 88–105.
40. May, A.; Ozerov, I. On computing nearest neighbors with applications to decoding of binary linear codes. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, 10–14 May 2020; Springer International Publishing: Berlin/Heidelberg, Germany, 2015; pp. 203–228.
41. Both, L.; May, A. Decoding linear codes with high error rate and its impact for LPN security. In Proceedings of the International Conference on Post-Quantum Cryptography, Fort Lauderdale, FL, USA, 9–11 April 2018; Springer International Publishing: Cham, Switzerland, 2018; pp. 25–46.
42. Esser, A.; Zweydinger, F. New time-memory trade-offs for subset sum–improving ISD in theory and practice. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, 10–14 May 2020; Springer Nature Switzerland: Cham, Switzerland, 2023; pp. 360–390.
43. Esser, A. Revisiting nearest-neighbor-based information set decoding. In Proceedings of the IMA International Conference on Cryptography and Coding, Oxford, UK, 14–16 December 2021; Springer Nature Switzerland: Cham, Switzerland, 2023; pp. 34–54.
44. Furue, H.; Aikawa, Y. An Improved Both-May Information Set Decoding Algorithm: Towards More Efficient Time-Memory Trade-Offs. In Proceedings of the International Conference on Post-Quantum Cryptography, Fort Lauderdale, FL, USA, 9–11 April 2018; Springer Nature Switzerland: Cham, Switzerland, 2025; pp. 104–128.
45. Bernstein, D.J. Grover vs. mceliece. In Proceedings of the International Workshop on Post-Quantum Cryptography, Utrecht, The Netherlands, 26–28 June 2017; Springer International Publishing: Berlin/Heidelberg, Germany, 2010; pp. 73–80.
46. Nguyen, P.Q.; Shparlinski, I.E.; Stern, J. Distribution of modular sums and the security of the server aided exponentiation. In *Cryptography and Computational Number Theory*; Birkhäuser: Basel, Switzerland, 2001; pp. 331–342.
47. Ambainis, A. Quantum walk algorithm for element distinctness. *SIAM J. Comput.* **2007**, *37*, 210–239. [CrossRef]
48. Grover, L.K. Quantum computers can search arbitrarily large databases by a single query. *Phys. Rev. Lett.* **1997**, *79*, 4709–4712. [CrossRef]
49. Helm, A.; May, A. The power of few qubits and collisions–subset sum below Grover's bound. In Proceedings of the International Conference on Post-Quantum Cryptography, Fort Lauderdale, FL, USA, 9–11 April 2018; Springer Nature Switzerland: Cham, Switzerland, 2020; pp. 445–460.
50. Chevignard, C.; Fouque, P.A.; Schrottenloher, A. Reducing the number of qubits in quantum information set decoding. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, 2–6 December 2018; Springer Nature Singapore: Singapore, 2024; pp. 299–329.