Improving Schroeppel and Shamir's Algorithm for Subset Sum via Orthogonal Vectors

Jesper Nederlof[*]
Utrecht University
Utrecht, The Netherlands
j.nederlof@uu.nl

Karol Węgrzycki[†]
Saarland University and Max Planck Institute for Informatics
Saarbrücken, Germany
wegrzycki@cs.uni-saarland.de

ABSTRACT

We present an $O^*(2^{0.5n})$ time and $O^*(2^{0.249999n})$ space randomized algorithm for solving worst-case Subset Sum instances with n integers. This is the first improvement over the long-standing $O^*(2^{n/2})$ time and $O^*(2^{n/4})$ space algorithm due to Schroeppel and Shamir (FOCS 1979).

We breach this gap in two steps: (1) We present a space efficient reduction to the Orthogonal Vectors Problem (OV), one of the most central problem in Fine-Grained Complexity. The reduction is established via an intricate combination of the method of Schroeppel and Shamir, and the representation technique introduced by Howgrave-Graham and Joux (EUROCRYPT 2010) for designing Subset Sum algorithms for the average case regime. (2) We provide an algorithm for OV that detects an orthogonal pair among N given vectors in $\{0,1\}^d$ with support size $d/4$ in time $\tilde{O}(N \cdot 2^d / \binom{d}{d/4})$. Our algorithm for OV is based on and refines the representative families framework developed by Fomin, Lokshtanov, Panolan and Saurabh (J. ACM 2016).

Our reduction uncovers a curious tight relation between Subset Sum and OV, because any improvement of our algorithm for OV would imply an improvement over the runtime of Schroeppel and Shamir, which is also a long standing open problem.

CCS CONCEPTS

• **Theory of computation → Parameterized complexity and exact algorithms**; *Computational complexity and cryptography*; *Algorithm design techniques.*

KEYWORDS

Knapsack, Subset Sum, Meet-in-the-Middle, Space Complexity, Representation Technique

ACM Reference Format:
Jesper Nederlof and Karol Węgrzycki. 2021. Improving Schroeppel and Shamir's Algorithm for Subset Sum via Orthogonal Vectors. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC '21), June 21–25, 2021, Virtual, Italy.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3406325.3451024

1 INTRODUCTION

The most natural question in computational complexity is: Can an algorithm be improved, or is there some fundamental barrier stopping us from doing so? A major theme in contemporary research has been to study this question in a *fine-grained* sense: given an algorithm using $\mathcal{T}$ time (and $\mathcal{S}$ space) on worst-case instances, can this be improved to $\mathcal{T}^{1-\varepsilon}$ (or $\mathcal{S}^{1-\varepsilon}$ space), for some $\varepsilon > 0$?

Because of the highly challenging nature of finding such improvements, researchers introduced several hypotheses that state that the currently best known algorithms already hit upon a barrier and therefore cannot be improved in the above sense. Under these hypotheses, many simple algorithms for standard problems in P like 3-SUM, Edit Distance, or Diameter cannot be significantly improved. The fine-grained hardness of the latter two problems is based on the hardness of a problem that is particularly central in the area, called *Orthogonal Vectors*: Given N vectors in $\{0,1\}^d$, detect two orthogonal vectors. A common hypothesis is that for $d = \omega(\log n)$ the problem cannot be solved in $O(N^{2-\varepsilon})$ time for some constant $\varepsilon > 0$. See for example the survey [38].

For NP-complete problems the situation is slightly different: Although similar fine-grained hypotheses for CNF-SAT and Set Cover have been introduced, they did not prove sufficient yet to rule out improvements of the currently best algorithms for basic NP-complete problems such as Traveling Salesman, Graph Coloring and MAX-3-SAT. See the survey [27] for some hardness results in this regime. There may be a good reason for this: While improved polynomial time algorithms can be naturally used as subroutines for improved exponential time algorithms, the converse is far less natural. Therefore it is quite plausible that finding better exponential time algorithms is much easier than finding faster polynomial time algorithms. And indeed, in the last decade improved algorithms for basic problems such as Undirected Hamiltonicity [11] and Graph Coloring [13] were found. This motivates the optimism that for many NP-complete problems the currently best known algorithms can still be enhanced.

[*] Supported by the project CRACKNP that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 853234).

[†] Supported by the project TIPEA that has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (grant agreement No. 850979). Author was also supported Foundation for Polish Science (FNP), by the grants 2016/21/N/ST6/01468 and 2018/28/T/ST6/00084 of the Polish National Science Center and project TOTAL that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 677651).

STOC '21, June 21–25, 2021, Virtual, Italy
© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-8053-9/21/06…$15.00
https://doi.org/10.1145/3406325.3451024

Equipped with this optimism, we study the fine-grained complexity of the following important class of NP-complete problems revolving around numbers.

Subset Sum, Knapsack and Binary Integer Programming. In the Subset Sum problem, we are given as input a set of integers $\{w_1, \ldots, w_n\}$ and a target t. The task is to decide if there exists a subset $S \subseteq \{1, \ldots, n\}$ such that the total sum of integers $w(S) := \sum_{i \in S} w_i$ is equal to t.

In the 1970's, Horowitz and Sahni [23] introduced the *meet-in-the-middle strategy* and solved Subset Sum in $O^*(2^{n/2})$ time and space. Since then, it has been a notorious open question to improve their result:

Question 1: Can Subset Sum be solved in $O^*(2^{(1/2-\varepsilon)n})$ time, for $\varepsilon > 0$?

A few years later, Schroeppel and Shamir [37] gave an algorithm for Subset Sum using $O^*(2^{n/2})$ time and only $O^*(2^{n/4})$ *space*. In the last section of their paper, they ask the following:

Question 2: Can Subset Sum be solved in $O^*(2^{n/2})$ time and $O^*(2^{(1/4-\varepsilon)n})$ space, for $\varepsilon > 0$?

Both questions seemed to be out of reach until 2010, when Howgrave-Graham and Joux [24] introduced the *representation technique* and used it to solve *random instances* of Subset Sum in $O^*(2^{0.337n})$ time. The main idea behind the representation technique is to artificially expand the search space such that a single solution has an exponential number r of representatives in the new search space. This allows us to subsequently restrict attention to a $1/r$-fraction of the search space, which in some settings can be advantageous. In the context of Subset Sum, this technique has already inspired improved algorithms for large classes of instances [7, 8], time-space trade-offs [6, 20] and improved polynomial space algorithms [9].

Nevertheless, answers to Questions 1 and 2 for worst-case instances still remained elusive.

1.1 Our Main Result and Key Insight

Our main result is a positive answer to the 40-year old open Question 2:

Theorem 1. *Every instance of Subset Sum can be solved in $O^*(2^{n/2})$ time and $O^*(2^{0.249999n})$ space by a randomized Monte Carlo algorithm with constant success probability.*

The result implies an analogous space improvement for Knapsack and Binary Integer Programming (see Corollary 4). To explain our key ideas and their combination with existing methods, the following problem is instrumental:

Weighted Orthogonal Vectors (notation: $\mathrm{WOV}(N, d, h)$)

Input: Families of N weighted sets $\mathcal{A}, \mathcal{B} \subseteq \binom{[d]}{h} \times \mathbb{N}$ of Hamming weight h, target integer t.

Task: Detect $(A, w_A) \in \mathcal{A}$ and $(B, w_B) \in \mathcal{B}$ such that A and B are disjoint and $w_A + w_B = t$.

The starting point is the $O^*(2^{n/2})$ time and space algorithm [23]. Their algorithm can be seen as a reduction to an instance of WOV($2^{n/2}$,0,0). Since $d = 0$, this is an instance of 2-SUM, and the runtime follows by a linear time algorithm for 2-SUM.

In contrast, the representation technique [24] can also be thought of as a reduction from instances of Subset Sum to WOV, but with the assumption that the Subset Sum instance does not have *additive structure.*[1] In a follow-up work, Austrin et al. [8] loosen the assumption of [24] and show that their reduction applies whenever there is a small subset of d weights without additive structure. Their work implies a reduction from every[2] instance of Subset Sum to $\mathrm{WOV}(N, d, d/4)$, where $N = 2^{n/2}\binom{d}{d/4}/2^d$ and $d/n > 0$ is a small (but fixed) positive constant.

Note that the two above reductions feature an intriguing trade-off between the *size* N and the *dimension* d of the produced instance, and the natural question is how the worst-case complexities of solving these instances as quick as possible compare. Our first step towards proving Theorem 1 is to show that this trade-off is **tight**, unless Question 1 is answered positively:

Key Insight: There is an algorithm for WOV whose run time dependency in d matches the instance decrease in d in the reduction from [8]. In particular, $\mathrm{WOV}(N, d, d/4)$ can be solved in $\tilde{O}(N \cdot 2^d/\binom{d}{d/4})$ time and $\tilde{O}(N + 2^d)$ space (see Theorem 28).

This insight has two interesting immediate consequences. First, it provides an avenue towards resolving Question 1, because a positive answer to this question is implied by an improvement of our algorithm even for the unweighted version of $\mathrm{WOV}(N, d, d/4)$. To the best of our knowledge, such an improvement is entirely consistent with all the known hypotheses on (low/moderate/sparse) versions of the Orthogonal Vectors problem [18, 22]. In fact, to answer Question 1 affirmatively we only need an improvement for the regime $2^d/\binom{d}{d/4} \leq N \leq 2^d$, while previous hypotheses address the regime where $d/\log N$ tends to infinity.

Second, a combination of the reduction from [8] and our algorithm for $\mathrm{WOV}(N, d, d/4)$ would give an algorithm for Subset Sum that runs in $O^*(2^{n/2})$ time and $O^*(2^{(1/2-\delta)n})$ space, for some small $\delta > 0$. While this is not even close to the memory improvement of [37], one may hope that by adding the ideas [37] on top of this approach results in a better memory usage. Notwithstanding the significant hurdles that need to be overcome to make these two methods combine, this is exactly how we get the improvement in Theorem 1.

1.2 The Representation Technique Meets Schroeppel and Shamir's Technique

We now give a high level proof idea of Theorem 1. While our conceptual contribution lies in the aforementioned key insight, our main *technical* effort lies in showing that indeed the representation technique and the algorithm of Schroeppel and Shamir [37] can be combined to get a space efficient reduction from Subset Sum to WOV.

[1] Specifically, this means that $|w(2^{[n]})| \geq 2^{(1-\varepsilon)n}$ for some small $\varepsilon > 0$, where $w(2^{[n]})$ denotes $\{w(X) : X \subseteq [n]\}$.

[2] Actually not every instance, but instances where the reduction fails can be solved quickly by other means.

The method of Schroeppel and Shamir [37] can also be seen as a reduction from Subset Sum to an instance of $\mathrm{WOV}(2^{n/2}, 0, 0)$, but it is an *implicit* one: The relevant vectors of the instance can be enumerated quickly by decomposing the search space of $2^{n/2}$ vectors into a Cartesian product of two sets of $2^{n/4}$ vectors, and generating all vectors in a useful order via priority queues. See Section 3 for a further explanation. Thus, to prove Theorem 1, we aim to generate the relevant parts of the instance $\mathcal{I}$ of $\mathrm{WOV}(N, d, d/4)$ defined by the representation technique efficiently, using priority queues of size at most $O^*(2^{0.249999n})$.

Unfortunately, the vectors from the instance $\mathcal{I}$ defined by the representation technique are elements of a search space of size $2^{(1/2+\Omega(1))n}$; its crux is that there are only $2^{(1/2-\Omega(1))n}$ vectors in the instance because we only have vectors with a fixed inner product with the weight vector $(w_1, \dots, w_n)$.[3] Thus a straightforward decomposition of this space into a Cartesian product will give priority queues of size $2^{(1/4+\Omega(1))n}$.

To circumvent this issue, we show that we can apply the representation technique *again* to generate the vectors of the instance $\mathcal{I}$ efficiently using priority queues of size $O^*(2^{0.249999n})$. While the representation technique was already used in a multi-level fashion in several earlier works (see e.g. [6, 24]), an important ingredient of our algorithm is that we apply the technique in different ways at the different levels depending on the structure of the instance.

1.3 Additional Results and Techniques

Our route towards Theorem 1 as outlined above has the following by-products that may be considered interesting on their own. The first one was already referred to in the 'key insight':

An algorithm for Orthogonal Vectors. A key subroutine in this paper is the following algorithm for Orthogonal Vectors. We let $\mathrm{OV}(N, d, h)$ refer to the problem $\mathrm{WOV}(N, d, h)$ restricted to unweighted instances (that is all involved integers are zero).

Theorem 2. *There is a Monte-Carlo algorithm for* $\mathrm{OV}(N, d, d/4)$ *that uses* $\tilde{O}\left(N \cdot 2^d / \binom{d}{d/4}\right)$ *time and* $\tilde{O}(N + 2^d)$ *space.*

An easy reduction shows the same runtime and space usage can be obtained for $\mathrm{WOV}(N, d, d/4)$. Our algorithm for Orthogonal Vectors uses the general blueprint of an algorithm by Fomin et al. [21] (which in turn builds upon ideas from [14, 29]). However, to ensure that the algorithm for Orthogonal Vectors combined with our methods result in an $O^*(2^{n/2})$ time algorithm for Subset Sum, we need to refine their method and analysis.

To facilitate our presentation, we consider a new communication complexity-related parameter of 1-*covers of a matrix* that we call the '*sparsity*'. We show that a 1-cover of low sparsity of a specific *Disjointness Matrix* implies an efficient algorithm for Orthogonal Vectors, and we exhibit a 1-cover of low sparsity of the disjointness matrix. We also prove that our 1-cover has nearly optimal sparsity. This means that Question 1 cannot be resolved directly via our route combined with improved 1-covers. Additionally, we use several preprocessing techniques to ensure that the space usage of our algorithm is only $\tilde{O}(N + 2^d)$, which is crucial to Theorem 1.

Reduction to weighted P_4. While we do not resolve Question 1, our approach provides new avenues by reducing it to typical questions in the study of fine-grained complexity of problems in the complexity class P. Our new reductions enable us to show a new connection between Subset Sum and the following graph problem: In the Exact Node Weighted P_4 problem one is given an undirected graph $G = (V, E)$ with vertex weights and the task is to decide whether there exists a path on four vertices with weights summing to 0 (see the full version of this paper [35]).

Theorem 3. *If Exact Node Weighted* P_4 *on a graph* $G = (V, E)$ *can be solved in* $O(|V|^{2.05})$ *time, then Subset Sum can be solved in* $O^*(2^{(0.5-\delta)n})$ *randomized time for some* $\delta > 0$.

In comparison to the straightforward reduction from Subset Sum to 4-SUM, our reduction creates a set of integers with an additional path constraint. Thus a possible attack towards resolving Question 1 is to design a quadratic time algorithm for Exact Node Weighted P_4 (or more particularly, only for the instances of the problem generated by our reduction).

The naïve algorithm for Exact Node Weighted P_4 works in $\tilde{O}(|V|^3)$ time. To the best of our knowledge the best algorithm for this problem runs in $\tilde{O}(|V|^{2.5})$ when $\omega = 2$ [16] (where ω is the exponent of currently the fastest algorithm for matrix multiplication).[4] On the lower bounds side, using the 'vertex minor' method from [3] it can be shown that the problem of detecting triangles in a graph can be reduced to Exact Node Weighted P_4 [1]. This explains that obtaining a quadratic time algorithm may be hard (since it is even hard to obtain for detecting triangles). However, detecting triangles in a graph is known to be solvable in $\tilde{O}(|V|^{\omega})$ time, Therefore it is still justified to aim for a $\tilde{O}(|V|^{\omega})$ time algorithm for Exact Node Weighted P_4. We leave it as an intriguing open question whether Exact Node Weighted P_4 can be solved in $\tilde{O}(|V|^{\omega})$ time.

More general problems. Known reductions from [34] combined with Theorem 1 also imply the following improved algorithms for generalizations of the Subset Sum problem (see the full version of this paper [35] for their definitions):

Corollary 4. *Any instance of Knapsack on n items can be solved in* $O^*(2^{n/2})$ *time and* $O^*(2^{0.249999n})$ *space, and any instance of Binary Integer Programming with n variables and d constraints with maximum absolute integer value m can be solved in* $O^*(2^{n/2}(\log(mn)n)^d)$ *time and* $O^*(2^{0.249999n})$ *space.*

1.4 Related Work

It was shown in [37] that Subset Sum admits a time-space tradeoff, i.e. an algorithm using $\mathcal{S}$ space and $2^n/\mathcal{S}^2$ time for any $\mathcal{S} \leq O^*(2^{n/4})$. This tradeoff was improved by [6] for almost all tradeoff parameters (see also [20]). We mention in the passing that as direct consequence of Theorem 1, the Subset Sum admits a time-space tradeoff using $2^n/\mathcal{S}^{0.5/0.249999}$ time and $\mathcal{S}$ space, for any

[3] Some knowledge of the representation technique is required to understand this in detail; We explain the representation technique in Section 2.

[4] Briefly described, reduce a problem instance on (V, E) to the problem of finding a triangle in an unweighted graph on $|V|^2$ edges: One vertex of the triangle represents the two extreme vertices of the path and the sum of the weights of the two first vertices on the path, and the other two vertices of the triangle represent the inner vertex. This instance of unweighted triangle can be solved in $\tilde{O}(|V|^{2.5})$ with standard methods, assuming $\omega = 2$.

$\mathcal{S} \le O^*(2^{0.249999n})$, but the obtained parameters are only better than the previous works for $\mathcal{S}$ chosen closely to its maximum. See the full version of this paper [35] for a proof.

In [7], the authors considered Subset Sum parametrized by the parameter β (which is defined as the largest number of subsets of the input integers that yield the same sum) and obtained an algorithm running in time $O^*(2^{0.3399n}\beta^4)$. Subsequently, [8] showed that one can get a faster algorithm for Subset Sum than meet-in-the-middle if $\beta \le 2^{(0.5-\varepsilon)n}$ or $\beta \ge 2^{0.661n}$. Recently, [9] gave an algorithm for Subset Sum running in $O^*(2^{0.86n})$ time and polynomial space, assuming random access to a random oracle.

From the pseudopolynomial algorithms perspective, Subset Sum has also been subject of recent stimulative research [2, 15, 19, 25, 26, 28]. These algorithms use plethora of ingenious algorithmic techniques, e.g., dynamic programming, color-coding and the Fast Fourier Transform.

Average case complexity and representation technique. In a breakthrough paper, Howgrave-Graham and Joux [24] introduce the *representation technique* and showed $O^*(2^{0.337n})$ time and $O^*(2^{0.256n})$ space algorithm for an Subset Sum in average-case setting. It was improved by [10] who gave an algorithm running in $O^*(2^{0.291n})$ time and space.

The representation technique already found several applications in the worst-case setting for other problems (see [30, 31, 33]).

Orthogonal Vectors. Naively, the Orthogonal Vectors problem can be solved in $O(dN^2)$ time. For large d, only a slightly faster algorithm that runs in time $N^{2-1/O(\log(d/\log N))}$ time is known [4, 17]. The assumption that for $d = \omega(\log N)$ there is no $O(N^{2-\varepsilon})$ time algorithm any $\varepsilon > 0$ is a central conjecture of fine-grained complexity (see [38] for an overview).

In this paper, we are mainly interested in linear (in N) time algorithms for OV. It was shown in [39] that OV cannot be solved in $O(N^{2-\varepsilon} \cdot 2^{o(d)})$ time for any $\varepsilon > 0$ assuming SETH. In [12] an algorithm for OV was given that runs in $\tilde{O}(D)$ time, where D is the total number of vectors whose support is a subset of the support of an input vector.

1.5 Organization

In this paper we heavily build upon previous literature, and in particular the representation technique as developed in [8, 10]. Therefore, we introduce the reader to this technique in Section 2. At the end of Section 2, we also use the introduced terminology of the representation technique to explain the new steps towards proving Theorem 1.

The remainder of the paper is devoted to formally support all claims made. Necessary preliminaries are provided in Section 3; in Section 4 we present the proof of Theorem 1, and Section 5 contains the proof of (a generalization of) Theorem 2.

In the full version of this paper [35] we provide various short omitted proofs, an inequality relevant for the runtime of our algorithms and the reduction to the Exact Node Weighted P_4 from Theorem 3.

2 INTRODUCTION TO THE REPRESENTATION TECHNIQUE, AND ITS EXTENSIONS

This section is devoted to explain the representation technique (and its extensions from [8]) and will serve towards a warm up towards the formal proof of Theorem 1.

2.1 The Representation Technique with a Simplified Assumption

We fix an instance $w_1, \dots, w_n, t$ of Subset Sum. A *perfect mixer* is a subset $M \subseteq [n]$ such that for every distinct subsets $A_1, A_2 \subseteq M$ we have $w(A_1) \neq w(A_2)$.[5] To simplify the explanation in this introductory section, we will make the following assumption about the Subset Sum instance:

Assumption 1. *If $w_1, \dots, w_n, t$ is a YES-instance of Subset Sum, then there is a perfect mixer $M \subseteq [n]$ and a set S with $w(S) = t$ such that $|M \cap S| = |M|/2$.*

A mild variant of Assumption 1 can be made without loss of generality since relatively standard extensions of the method by [37] can be used to solve the instance more efficiently if it does not hold. We discuss the justification of Assumption 1 more later. We now illustrate the representation technique by outlining proof of the following statement:

Theorem 5. *An instance of Subset Sum satisfying Assumption 1 can be reduced to an equivalent instance of*

$$\mathrm{WOV}\left(2^{n/2}\binom{|M|}{|M|/4}/2^{|M|}, |M|, |M|/4\right)$$

in the linear (in the size of the output) randomized time.

```
Algorithm : RepTechnique($w_1, \dots, w_n, t, M$)
Output    : Instance of weighted orthogonal vectors
Arbitrarily partition $[n] \setminus M$ into $L$ and $R$ such that
  $|L| = |R| = (n - |M|)/2$
Pick a random prime $p$ of order $2^{|M|/2}$
Pick a random $x \in \mathbb{Z}_p$
Construct the following sets:
    $\mathcal{L} := \{(A_1 \cap M, w(A_1))$ such that $A_1 \in 2^{L \cup M}$
                and $|A_1 \cap M| = |M|/4$ and $w(A_1) \equiv_p x\}$
    $\mathcal{R} := \{(A_2 \cap M, w(A_2))$ such that $A_2 \in 2^{R \cup M}$
                and $|A_2 \cap M| = |M|/4$ and $w(A_2) \equiv_p t - x\}$
return the instance $(\mathcal{L}, \mathcal{R}, t)$ of Weighted OV
```

Algorithm 1: Pseudocode of Theorem 5

The reduction from Theorem 5 is described in Algorithm 1, and uses the standard notation $\equiv_p$ to denote equivalence modulo p. We now describe the intuition of the algorithm. For partition of $[n]$ into L, M, R, it *expands* the search space by looking for pairs

[5] This notion will be generalized to the notion of an ε-mixer in Definition 11.

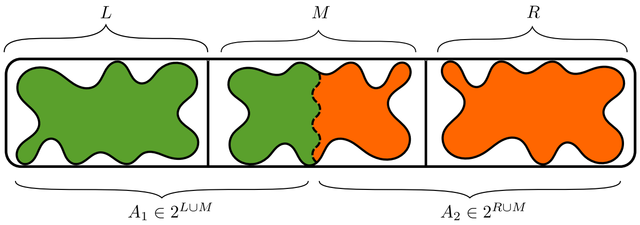

Figure 1: The green and orange regions represent a solution, i.e., a set S such that $w(S) = t$. There are $\binom{|M \cap S|}{|M \cap S|/2}$ pairs $A_1 \in 2^{L \cup M}$ and $A_2 \in 2^{R \cup M}$, such that $A_1 \cup A_2 = S$ and $A_1 \cap A_2 = \emptyset$. Because M is a perfect mixer, the number of possible values that $w(A_1 \cap M)$ can take is also $\binom{|M \cap S|}{|M \cap S|/2}$.

(A_1, A_2) where $A_1 \in 2^{L \cup M}$, $A_2 \in 2^{R \cup M}$ and both A_1 and A_2 use $|M|/4$ elements of M. This is useful since the assumed solution S is *represented* by the $\binom{|M \cap S|}{|M|/4} \approx 2^{|M|/2}$ partitions (A_1, A_2) of S that are in the expanded search space. Together with Assumption 1, this allows us in turn to narrow down the search space by restricting the search to look only for pairs (A_1, A_2) satisfying $w(A_1) \equiv_p x$, and thus $w(A_2) \equiv_p t - x$. Thus, the algorithm enumerates all candidates for A_1 and A_2 in respectively $\mathcal{L}$ and $\mathcal{R}$ and the instance of weighted orthogonal vectors detects a disjoint pair of candidates with weights summing to t.

One direction of the correctness of the algorithm follows directly: If the produced instance of weighted orthogonal vectors is a YES-instance, the union of the two found sets is a solution to the Subset Sum instance.

Conversely, we claim that if the instance of Subset Sum is a YES-instance and Assumption 1 holds, then with good probability the output instance of Weighted Orthogonal Vectors is a YES-instance. Let S be the solution of the Subset Sum instance, so $w(S) = t$ and $|S \cap M| = |M/2|$ by Assumption 1. Note that

$$W := \left|\left\{w(\tilde{A}_1 \cup (L \cap S)) : \tilde{A}_1 \in \binom{M \cap S}{|M \cap S|/2}\right\}\right| = \binom{|M|/2}{|M|/4},$$

because there are $\binom{|M|/2}{|M|/4}$ possibilities for $\tilde{A}_1$ and $w(\tilde{A}_1)$ is different for each different $\tilde{A}_1$ by the perfect mixer property of M. Therefore, there are $\binom{|M|/2}{|M/4}$ possibilities for $A_1 := A'_1 \cup (L \cap S)$ and each $w(A_1)$ is different.

By standard properties of hashing modulo random prime numbers (see e.g. Lemma 7 for a general statement), we have that the expected size of $\{y \mod p : y \in W\}$ is also approximately of cardinality $\binom{|M|/2}{|M|/4}$.[6] Therefore the probability that x is chosen such that $x \equiv_p y$ for some $y \in W$ is $\binom{|M|/2}{|M|/4}/2^{|M|/2} \geq \Omega(\frac{1}{|M|})$. If we let $A_1 \in \mathcal{L}$ be the set with $w(A_1) = y$ then since $w(S \setminus A_1) \equiv_p t - y$, $S \setminus A_1 \in \mathcal{R}$ and the pair $(A_1, S \setminus A_2)$ is a solution to the weighted orthogonal vectors problem. In general this happens with probability at least $1/n$.

Now we discuss the runtime and output size. At Line 4 we construct $\mathcal{L}$ and $\mathcal{R}$. Since the number of possibilities of A_1 is $2^{|L|}\binom{|M|}{|M|/4}$ and each such A_1 satisfies $w(A_1) \equiv_p x$ with probability p (taken over the random choices of x), we have that the expected sizes of $\mathcal{L}$ (and similarly, of $\mathcal{R}$) is $2^{n/2}\binom{|M|}{|M|/4}/2^{|M|}$, as claimed. By standard pseudo-polynomial dynamic programming techniques (see e.g. [8]), Line 4 can be performed in $O(p + |\mathcal{L}| + |\mathcal{R}|)$ time, and thus the claimed run time follows.

2.2 Representation Technique with Non-Simplified Assumption

Assumption 1 is oversimplifying our actual assumptions, and actually only a weaker assumption is needed to apply the representation method. We call any set M that satisfies $|w(2^M)| = 2^{(1-\varepsilon)|M|}$ an ε-*mixer* (see also Definition 11). Denoting $w(\mathcal{F})$ for $\{w(X) : X \in \mathcal{F}\}$, the assumption is

Assumption 2. *If $w_1, \ldots, w_n, t$ is a YES-instance of Subset Sum, then there is an ε-mixer M, and a set S with $w(S) = t$ such that $|(\frac{1}{2} - \varepsilon')|M| \leq |M \cap S| \leq (\frac{1}{2} + \varepsilon')|M|$, for some small positive $\varepsilon, \varepsilon'$.*

To note that the representation technique introduced above still works with these relaxed assumptions, we remark that it can be shown that if, M is an ε-mixer, then $w(\binom{M \cap S}{i}) \geq 2^{(1-f(\varepsilon,\varepsilon'))|M \cap S|}$ for some $\varepsilon, \varepsilon'$ and unknown i. Thus in the representation technique we can split the solution in sets A_1, A_2 where $|A_1 \cap M| = i$ and $|A_2 \cap M| = |S \cap M| - i$ and use a prime p of order $2^{(1-f(\varepsilon,\varepsilon'))|M \cap S|}$ for some function f that tends to 0 when ε and ε' tend to 0.

The advantage of the relaxed assumptions in Assumption 2 is that the methods from [23, 37] can be extended such that it solves any instance that does *not* satisfy the assumptions exponentially better in terms of time and space than in the worst-case. This allows us to make these assumptions without loss of generality when aiming for general exponential improvements in the run time (or space bound).

For example, in the approach by Schroeppel and Shamir [37], in some steps of the algorithm we only need to enumerate subsets of cardinality bounded away from half of the underlying universe; or in some other steps of the algorithm we can maintain smaller lists with sums generated by subsets. While these extensions are not entirely direct, we skip a detailed explanation of them in this introductory section (see Section 3 for formal statements).

2.3 Our Extensions of the Representation Technique Towards Theorem 1

Having described the representation technique, we now explain our approach in more detail.

Setting up the representation technique to reduce the space usage. Now, we present the intuition behind the space reduction of Theorem 1. In the previous subsection we constructed an instance $\mathcal{L}, \mathcal{R}$ of weighted orthogonal vectors of expected size $O^*(2^{n/2-\Omega(|M|)})$ such that with good probability there exist $A_1 \in \mathcal{L}$ and $A_2 \in \mathcal{R}$ with $w(A_1 \cup A_2) = t$ and $A_1 \cap A_2 = \emptyset$ (if the answer to Subset Sum is *yes*). We combine this approach with the approach from [37] and aim to efficiently enumerate this instance of weighted orthogonal vectors instance. To do so, we apply the representation method two times more, and are able to construct 4 sets $\mathcal{L}_1, \mathcal{L}_2, \mathcal{R}_1, \mathcal{R}_2 \subseteq 2^{[n]}$ with the following properties:

[6] This uses that all numbers are single-exponential in n, but this can be assured with a standard hashing argument.

(i) With good probability there exist pairwise disjoint $A_1 \in \mathcal{L}_1, A_2 \in \mathcal{L}_2, A_3 \in \mathcal{R}_2, A_4 \in \mathcal{R}_1$, such that $w(A_1 \cup A_2 \cup A_3 \cup A_4) = t$, if the Subset Sum instance is a YES-instance,

(ii) The expected size of each $\mathcal{L}_1, \mathcal{L}_2, \mathcal{R}_1, \mathcal{R}_2$ is $O^*(2^{n/4-\Omega(|M|)})$.

The sets will have the property that elements in $\mathcal{L}$ are formed by pairs from $\mathcal{L}_1 \times \mathcal{L}_2$ and elements in $\mathcal{R}$ are formed by pairs from $\mathcal{R}_1 \times \mathcal{R}_2$. But in contrast to the technique from [37], the lists $\mathcal{L}$ and $\mathcal{R}$ can not be easily decomposed into a Cartesian product of two sets of size $\sqrt{|\mathcal{L}|}$ and $\sqrt{|\mathcal{R}|}$. To overcome this issue, we apply the representation method *again* to enumerate the elements of $\mathcal{L}$ and $\mathcal{R}$ quickly. In particular, to construct the sets $\mathcal{L}_1, \mathcal{L}_2, \mathcal{R}_1, \mathcal{R}_2$, we partition the instance into L, M_L, M, M_R, R. See also Figure 2. Here M is assumed to be an ε-mixer that is used for the representation technique on 'first level': at this level we check whether a pair $\mathcal{L} \times \mathcal{R}$ forms a solution. The set M_L is assumed to be an ε_L-mixer and the set M_R is assumed to be an ε_R-mixer, and these sets are used for two applications of the representation technique on the 'second level': At this level we check whether a pair in $\mathcal{L}_1 \times \mathcal{L}_2$ forms an element of $\mathcal{L}$ (and similarly, whether a pair in $\mathcal{R}_1 \times \mathcal{R}_2$ forms an element of $\mathcal{R}$). If any of the assumptions fail, relatively direct extensions of the methods from [37] can again solve the instance more efficiently in a way similar to how we justified Assumption 2, so these assumptions are without loss of generality.

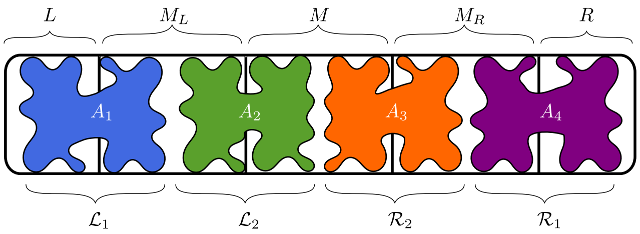

Figure 2: The decomposition of the instance into $[n] = L \uplus M_L \uplus M \uplus M_R \uplus R$ and decomposition of the solution $S = A_1 \uplus A_2 \uplus A_3 \uplus A_4$.

Maintaining the $O^*(2^{n/2})$ time bound. After we construct sets $\mathcal{L}_1, \mathcal{L}_2, \mathcal{R}_2, \mathcal{R}_1$ as claimed in property (i) we combine the approach of [37] with our Orthogonal Vectors algorithm to obtain the $O^*(2^{n/2})$ running time. In particular, we store the elements of $\mathcal{L}_1, \mathcal{L}_2, \mathcal{R}_1, \mathcal{R}_2$ in priority queues ordered by the weight in order to enumerate the elements of $\mathcal{L}$ and $\mathcal{R}$ in the correct order. By our applications of the representation technique, we only need to assure disjointness between pairs between $\mathcal{L}_1 \times \mathcal{L}_2, \mathcal{R}_1 \times \mathcal{R}_2$ and $\mathcal{L}_2 \times \mathcal{R}_2$ and therefore the time to check disjointness is the same as in the normal application of the representation technique as described at the beginning of this section.

Unfortunately, the relaxed assumptions from Assumption 2 cause issues here because we need to consider unbalanced partitions of $M \cap S$, $M_L \cap S$, $M_R \cap S$ and the constants $\varepsilon, \varepsilon_L, \varepsilon_M$ give rise to different primes in our application of the representation technique. Without additional care, the overhead in the runtime implied by these issues would lead to an undesired time bound of $O^*(2^{(0.5+\varepsilon)n})$ for arbitrarily small constant $\varepsilon > 0$.

To address these complications, we analyse our algorithm in such a way that if $|w(2^{M_L})|$ or $|w(2^{M_R})|$ is significantly smaller than $|w(2^M)|$, then we get an improved runtime. Note this can be assumed by switching the roles of M_L, M_R, M. Additionally, we provide a general runtime for solving Weighted Orthogonal Vectors instances with vectors with general support size.

3 PRELIMINARIES

Throughout the paper we use the $O^*(\cdot)$ notation to hide factors polynomial in the input size and the $\tilde{O}(\cdot)$ notation to hide polylogarithmic factors in the input size; which input this refers to will always be clear from the context. We also use $[n]$ to denote the set $\{1, \ldots, n\}$. We use the binomial coefficient notation for sets, i.e., for a set S the symbol $\binom{S}{k}$ denotes the set of all subsets of the set S of size exactly k. For a modulus $m \in \mathbb{Z}_{\geq 1}$ and $x, y \in \mathbb{Z}$ we write $x \equiv_m y$ to indicate that m divides $x - y$. If $X \subseteq [n]$, we denote $w(X) := \sum_{i \in X} w_i$, which is extended to set families $\mathcal{F} \subseteq 2^{[n]}$ by denoting $w(\mathcal{F}) := \{w(X) : X \in \mathcal{F}\}$. We use $A \uplus B = C$ to denote that A, B form a partition of C.

Prime numbers and hashing. We use the following folklore theorem on prime numbers:

Lemma 6 (Folklore)**.** *For every sufficiently large integer r the following holds. If p is a prime between r and $2r$ selected uniformly at random and x is a nonzero integer, then p divides x with probability at most $(\log_2 x)/r$.*

The following Lemma already appeared in [8], but since we need slightly different parameters we repeat its proof.

Lemma 7 (cf., Proposition 3.5 in [8])**.** *Let $w_1, \ldots, w_n$ be n integers bounded by $2^{O(n)}$. Suppose $Q \subseteq [n]$ with $|Q| = \Theta(n)$. Let $W_1, \ldots, W_c$ be integers and let $W = \prod_{i=1}^{c} W_i$ such that $W \leq |w(2^Q)|$. For $i = 1, \ldots, c$, let p_i be prime numbers selected uniformly at random from $[W_i/2, W_i]$. Let s_0 be the smallest integer such that $\binom{|Q|}{s_0} \geq |w(2^Q)|/|Q|$. Denoting $p := \prod_{i=1}^{c} p_i$, we have*

$$\mathbb{P}\left[\left|\left\{a \bmod p : X \subseteq Q,\ |X| \in [s_0, |Q|/2],\ w(X) = a\right\}\right| \geq \Omega\left(\frac{p}{n^c}\right)\right]$$

is greater or equal to 0.9*.*

We present the technical proof of Lemma 7 in the full version of this paper [35].

Shroeppel-Shamir's sumset enumeration. We recall some of the basic building blocks of previous work on Subset Sum. In [37] the authors used the following data structure to obtain an $\tilde{O}(n^2)$ time and $\tilde{O}(n)$ space algorithm for 4-SUM.

Lemma 8. *Let $A, B \subseteq \mathbb{Z}$ be two sets of integers, and let $C := A + B := \{a + b : a \in A, b \in B\}$ be their sumset. Let $c_1, \ldots, c_m$ be elements of C in increasing order. There is a data structure* `inc` *:=* `inc`*(A, B) that takes $\tilde{O}(|A|+|B|)$ preprocessing time and supports a query* `inc.next()` *that in the i'th (for $1 \leq i \leq m$) call outputs $(P^l_{c_i}, c_i)$, and in the subsequent calls outputs* `EMPTY`*. Here $P^l_{c_i}$ is the set $\{(a, b) : a \in A, b \in B, a + b = c_i\}$.*

Moreover, the total time needed to execute all m calls to `inc.next()` *is $\tilde{O}(|A||B|)$ and the maximum space usage of the data structure is $\tilde{O}(|A| + |B|)$.*

Similarly, there is a data structure `dec` *:=* `dec`*(A, B) that outputs pairs of elements of A and B in order of their decreasing sum.*

The data structure crucially relies on priority queues. We included the proof of this Lemma in the full version of this paper [35].

Binomial coefficients. We will frequently use the binary entropy function $h(p) := -p\log_2(p) - (1-p)\log_2(1-p)$. Its main use is via the following estimate of binomial coefficients:

$$\Omega(d^{-1/2})2^{dh(\alpha)} \leq \binom{d}{\alpha d} \leq 2^{dh(\alpha)}. \qquad (1)$$

We also consider the inverse of the binary entropy. Since $h(\alpha)$ is strictly increasing in $[0, 0.5]$ we can define $h^{-1} : [0, 1] \rightarrow [0, 0.5]$, with condition that $h^{-1}(\alpha) = \beta$ iff $h(\beta) = \alpha$.

For every $\alpha \in [0, 0.5]$ we have the following inequality on the entropy function:

$$1 - 4\alpha^2 \leq h(1/2 - \alpha) \leq 1 - 2\alpha^2/\ln 2 \qquad (2)$$

Moreover by the concavity of binary entropy we know that for all $\alpha, x, y \in [0, 1]$:

$$\alpha h(x) + (1-\alpha)h(y) \leq h(\alpha x + (1-\alpha)y) \qquad (3)$$

In particular it means that $h(\sigma\lambda) + h((1-\sigma)\lambda) \leq 2h(\lambda/2)$ for any $0 \leq \sigma \leq 1$.

Our runtime analysis crucially relies on the following highly technical inequality on binomial coefficients.

Lemma 9. *For large enough n and $\lambda \in [0.4, 0.5]$ and $\sigma \in [0.4, 0.6]$ the following inequality holds:*

$$\min_x \left\{ \frac{\binom{1-\lambda\sigma}{x-\lambda\sigma}_n + \binom{1-(1-\sigma)\lambda}{x}_n}{\binom{1-\lambda}{x-\lambda\sigma}_n} \right\} \leq 2^{n(1/2+\lambda-h(\lambda/2))} n^{O(1)}.$$

The proof of Lemma 9 is provided in the fullversion of this paper [35].

The following standard concentration lemma will be useful to control the intersection of the solution with certain subsets of the weights of the subset sum instance:

Lemma 10. *Let $A \subseteq [d]$ be any set with $|A| = \alpha d$, and let $B \subseteq [d]$ be uniformly sampled over all subsets with $|B| = \beta d$ and $\alpha\beta d$ be an integer. Then the following holds:*

$$\mathbb{P}\left[|A \cap B| = \alpha\beta d\right] \geq \Omega^*(1).$$

We provide the proof of Lemma 10 in the full version of this paper [35].

Preprocessing algorithms. We now present several simple procedures that allow us to make assumptions about the given Subset Sum instance in the proof of Theorem 1. Throughout this paper $w_1, \ldots, w_n, t$ denotes an instance of Subset Sum. We can assume that the integers $w_1, \ldots, w_n, t$ are positive and $w_1 + \ldots + w_n + t \leq 2^{10n}$ (see [8, Lemma 2.1]). Throughout the paper we will introduce certain constants close to 0 and assume that n is big enough, so the product of n with these constants is an integer.

The following notion that was already discussed in Section 2 corresponds to the number of distinct sums of the subsets of a given set.

Definition 11 (ε-mixer)**.** *A set $M \subseteq [n]$ is an* ε-mixer *if* $|w(2^M)| = 2^{(1-\varepsilon)|M|}$.

Lemma 12. *Given a set M, one can in $O^*(2^{|M|})$ time and $O^*(2^{|M|})$ space determine the ε such that M is an ε-mixer.*

Proof. Iterate over every possible subset of M and store $w(2^M)$. Afterwards sort $w(2^M)$, determine the size of M and output $\varepsilon := (1 - \log_2(|w(2^M)|)/|M|)$. □

Lemma 13. *For any constants $\varepsilon_0 > 0$ and $\mu \in (0, 1/4)$, there is an algorithm that, given a Subset Sum instance $w_1, \ldots, w_n, t$ and an ε-mixer M satisfying $|M| = \mu n$ and $\varepsilon > \varepsilon_0$, solves the instance in time $O^*(2^{(1-\varepsilon_0\mu)n/2})$ and $O^*(2^{(1-\varepsilon_0\mu)n/4})$ space.*

Lemma 14. *Suppose a Subset Sum instance $w_1, \ldots, w_n, t$ with promise that there is a solution of size λn is given. Then we can find $S \subseteq [n]$ with $w(S) = t$ in randomized $O^*(2^{h(\lambda)n/2} + 2^{n/4})$ time and $O^*(2^{h(\lambda)n/4})$ space.*

The proofs of Lemma 13 and Lemma 14 are a straightforward application of the algorithm for 4-SUM and we defer them to the full version of this paper [35].

4 IMPROVING SCHROEPPEL AND SHAMIR: PROOF OF THEOREM 1

This section is devoted to the proof of Theorem 1. The main technical effort, done in Subsections 4.1 to 4.2, is to prove the following lemma.

Lemma 15 (Main Lemma)**.** *Let $\lambda_0 := 0.495$, $\varepsilon_0 := 0.00002$. Let $\lambda \in [\lambda_0, 0.5]$, $\varepsilon_R \in [0, \varepsilon_0]$, $\mu \in (0.21, 0.25)$ and let $M_L, M, M_R \subseteq [n]$ be disjoint sets such that $|M| = |M_L| = |M_R| = \mu n$. Let $0 \leq \varepsilon \leq \varepsilon_L \leq \varepsilon_R$ be such that M_L is an ε_L-mixer, M is an ε-mixer and M_R is an ε_R-mixer. Let $S \subseteq [n]$ be such that $w(S) = t$ and $|M_L \cap S| = |M \cap S| = |M_R \cap S| = \lambda\mu n$.*

There is a Monte Carlo algorithm for Subset Sum that, given the instance $w_1, \ldots, w_n, t$, the sets M_L, M, M_R, and $\lambda, \varepsilon_L, \varepsilon_R$, runs in time $O^*(2^{n/2})$ and space

$$O^*\left(2^{(1/2-\mu(3/2+\lambda-h(1/4)))n+0.02\mu n} + 2^{\mu n(2h(1/4)-\lambda)+0.02\mu n} + 2^{\mu n}\right).$$

The performance of the algorithm depends on the parameters λ, μ, ε_L and ε_R. It is instructive to think about $\varepsilon_L = \varepsilon_R = 0$ and $\lambda = 1/2$.

First, we prove the main result of the paper assuming Lemma 15 by using the elementary preprocessing algorithms provided in Section 3.

Proof Theorem 1 assuming Lemma 15. Set $\mu := 0.217$. With polynomial overhead we can guess $|S| = \lambda n$. If $\lambda < \lambda_0$ then we use Theorem 14 to solve Subset Sum in $O^*(2^{h(\lambda)n/4}) \leq O^*(2^{0.249982n})$ space and $O^*(2^{n/2})$ time. Hence, we can assume that $\lambda \geq \lambda_0$. We can also assume that $\lambda \leq 1/2$ by looking for $[n] \setminus S$ instead of S by changing t to $w([n]) - t$.

Next, randomly select pairwise disjoint sets $M, M_L, M_R \in \binom{[n]}{\mu n}$. For each of them we use Lemma 12 to determine the $\varepsilon, \varepsilon_L, \varepsilon_R$ such that M is an ε-mixer, M_L is an ε_L-mixer and M_R is an ε_R-mixer. If at least one of $\varepsilon, \varepsilon_L, \varepsilon_R$ is at least ε_0, use Theorem 13 to solve the instance in $O^*(2^{(1-\mu\varepsilon_0)n/4}) \leq O^*(2^{0.24999892})$ space and $O^*(2^{n/2})$ time. Hence we can assume $\varepsilon, \varepsilon_L, \varepsilon_R < \varepsilon_0$.

Finally, Lemma 15 applies and it solves the instance in time $O^*(2^{n/2})$. For our choice of the parameters we get that the space is at most $O^*(2^{0.2491n})$.

In total, the space complexity of our algorithm is bounded by $O^*(2^{0.249999n})$ as claimed. □

The rest of this section is devoted to the proof of Lemma 15. This lemma is an extension of Theorem 5 combined with a fast OV algorithm. As mentioned in Subsection 2.3, we apply the representation technique on 2 levels and therefore we need 3 sets M_L, M, M_R. Moreover, the assumption $0 < \varepsilon \leq \varepsilon_L, \varepsilon_R$ is to avoid the aforementioned undesired $O^*(2^{(0.5+O(\varepsilon))n})$ running time.

4.1 The Algorithm for Lemma 15

```
   Algorithm : SubsetSum($w_1, \dots, w_n, t, M_L, M, M_R, \lambda, \varepsilon_L, \varepsilon_R$)
   Output    : Set $S$ with $w(S) = t$ and $|M_L \cap S|, |M \cap S|$,
               $|M_R \cap S| = \lambda|M|$, if it exists
1  Partition $[n] \setminus (M_L \cup M \cup M_R) = L \uplus R$ to satisfy (5)
2  Pick random prime $p_R \in \Theta(2^{(\lambda-\varepsilon_R)|M|})$
3  Pick random prime $p' \in \Theta(2^{(\varepsilon_R-\varepsilon_L)|M|})$
4  Set $p_L = p' \cdot p_R$
5  Pick random $x_L \in \mathbb{Z}_{p_L}, x \in \mathbb{Z}_{p_L}, x_R \in \mathbb{Z}_{p_R}$
6  foreach $\sigma, \sigma_L, \sigma_R$ with bounded $h(\sigma), h(\sigma_L), h(\sigma_R)$ do
7      Construct $\mathcal{L}_1, \mathcal{L}_2, \mathcal{R}_1, \mathcal{R}_2$ as in Equations (6) to (9)
8      if WeightedOV($\mathcal{L}_1, \mathcal{L}_2, \mathcal{R}_1, \mathcal{R}_2, M, t$) then
9          return true
10 return false
```

Algorithm 2: Pseudocode of the algorithm for Lemma 15

Algorithm 2 presents the pseudocode of Lemma 15. The WeightedOV subroutine decides whether there exists $(A_1, \dots, A_4) \in \mathcal{L}_1 \times \mathcal{L}_2 \times \mathcal{R}_1 \times \mathcal{R}_2$ with $w(A_1 \cup \dots \cup A_4) = t$ and $A_i \cap A_j = \emptyset$ for all $i \neq j$. This subroutine will be provided and analysed later in the Section 4.2.

On a high level, Algorithm 2 has the same structure as Algorithm 1, with one major difference: The sets $\mathcal{L}$ and $\mathcal{R}$ are generated implicitly. To generate these lists we combine the technique from [37] as summarized in Lemma 8 with two more applications of the representation technique used to generate $\mathcal{L}$ and $\mathcal{R}$.[7]

The algorithm iterates over every possible choice of parameters $\sigma, \sigma_L, \sigma_R \in [0, 1]$, such that $h(\sigma), h(\sigma_L) \geq 1 - \varepsilon_L/\lambda - \frac{\log_2 n}{n}$ and $h(\sigma_R) \geq 1 - \varepsilon_R/\lambda - \frac{\log_2 n}{n}$ in Line 6. The precision of $\sigma, \sigma_R, \sigma_L$ is polynomial, since these parameter describe the size of possible subsets of M, M_R, M_L. The purpose of one iteration of this loop is summarized in the following lemma, which is also illustrated in Figure 3:

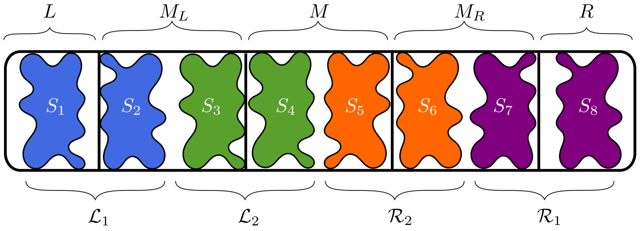

Figure 3: The decomposition of the solution $S = S_1 \uplus \dots \uplus S_8$ as formalized in Lemma 16.

Lemma 16. *Consider an iteration of the loop at Line 6 of Algorithm 2 with parameters $\sigma, \sigma_L, \sigma_R$. Suppose there exists a set $S \in \binom{[n]}{\lambda n}$ with $w(S) = t$ that has a partition $S = S_1 \uplus S_2 \uplus \cdots \uplus S_8$ satisfying the following properties:*

$$\begin{aligned} &S_1 \subseteq L, && S_2 \in \binom{M_L}{\sigma_L \lambda |M|}, \\ &S_8 \subseteq R, && S_7 \in \binom{M_R}{\sigma_R \lambda |M|}, \\ &S_3 \in \binom{M_L}{(1-\sigma_L)\lambda|M|}, && S_4 \in \binom{M}{\sigma\lambda|M|}, \\ &S_6 \in \binom{M_R}{(1-\sigma_R)\lambda|M|}, && S_5 \in \binom{M}{(1-\sigma)\lambda|M|}, \end{aligned} \tag{4}$$

$$w(S_1 \cup S_2) \equiv_{p_L} x_L, \quad w(S_3 \cup S_4) \equiv_{p_L} x - x_L,$$
$$w(S_5 \cup S_6) \equiv_{p_R} x_R, \quad w(S_7 \cup S_8) \equiv_{p_R} t - x - x_R,$$
$$w(S_1 \cup S_2 \cup S_3 \cup S_4) \equiv_{p_L} x,$$
$$w(S_5 \cup S_6 \cup S_7 \cup S_8) \equiv_{p_R} t - x.$$

Then during this iteration the Algorithm 2 returns true.

The (relatively straightforward) proof of Lemma 16 will be given in Subsection 4.3 where we prove the correctness of the algorithm. To obtain a relatively fast algorithm in the case that ε is bounded away from 0 or λ is bounded away from $1/2$, we need to carefully define the lists $\mathcal{L}_1, \mathcal{L}_2, \mathcal{R}_1, \mathcal{R}_2$ in order to not slow down the run time to beyond $O^*(2^{n/2})$. To do so, we use the following balance parameter

$$\beta = \beta(\lambda, \sigma) := h(\sigma\lambda) - h((1-\sigma)\lambda).$$

Intuitively, β expresses the difference of the expected list sizes $\{\mathcal{L}(a)\}_a$ and $\{\mathcal{R}(b)\}_b$ when we would have set $|L| = |R|$. Observe that if $\varepsilon = 0$ and $\lambda = 1/2$, then $\sigma_L, \sigma, \sigma_R = 1/2$ and indeed $\beta = 0$.

All elements of $[n]$ not in $M_L \cup M \cup M_R$ are arbitrarily partitioned into L and R on Line 1 where $|L|$ and $|R|$ are chosen to compensate for imbalance caused by $\varepsilon, \sigma, \lambda$ as follows:

$$|L| = \frac{(1 - 3\mu - \beta\mu)n}{2}, \qquad |R| = \frac{(1 - 3\mu + \beta\mu)n}{2}. \tag{5}$$

Observe that $|\beta| \leq 1$, and since $\mu \leq 1/4$ we have that $|L|, |R| > 0$.

Now we define the four lists that play a similar role in our algorithm as the four lists in the original algorithm of [37].

$$\mathcal{L}_1 := \Big\{ S_1 \cup S_2 \text{ such that } w(S_1 \cup S_2) \equiv_{p_L} x_L, \tag{6}$$
$$S_1 \subseteq L, S_2 \in \binom{M_L}{\sigma_L \lambda |M|} \Big\},$$

[7] Note that, formally speaking, the list $\mathcal{L}$ from Algorithm 1 is not the same as the set of elements of list $\mathcal{L}$ of Algorithm 2, but since the two are almost identical we kept the same notation.

$$\mathcal{R}_1 := \Bigg\{ S_7 \cup S_8 \text{ such that } S_8 \subseteq R,\ S_7 \in \binom{M_R}{\sigma_R \lambda |M|}, \quad w(S_7 \cup S_8) \equiv_{p_R} t - x - x_R \Bigg\}, \tag{7}$$

$$\mathcal{L}_2 := \Bigg\{ S_3 \cup S_4 \text{ such that } S_3 \in \binom{M_L}{(1-\sigma_L)\lambda|M|}, \quad S_4 \in \binom{M}{\sigma\lambda|M|}, w(S_3 \cup S_4) \equiv_{p_L} x - x_L \Bigg\}, \tag{8}$$

$$\mathcal{R}_2 := \Bigg\{ S_5 \cup S_6 \text{ such that } w(S_5 \cup S_6) \equiv_{p_R} x_R, \quad S_5 \in \binom{M}{(1-\sigma)\lambda|M|}, S_6 \in \binom{M_R}{(1-\sigma_R)\lambda|M|} \Bigg\}. \tag{9}$$

Using a straightforward algorithm, we can construct each list using $\tilde{O}(|\mathcal{L}_1| + |\mathcal{L}_2| + |\mathcal{R}_1| + |\mathcal{R}_2| + 2^{\mu n})$ time and space (see the full version of this paper [35]).

4.2 The Weighted Orthogonal Vectors Subroutine

Now we describe the WeightedOV subroutine (in the full-version of this paper [35] we provide the pseudocode). The algorithm is heavily based on the data structures from [37] as described in Lemma 8. First we initialize the queue inc for enumerating $w(\mathcal{L}_1) + w(\mathcal{L}_2)$ in the increasing order and the queue dec for enumerating $w(\mathcal{R}_1)+w(\mathcal{R}_2)$ in the decreasing order. With these queues, we enumerate all groups $\mathcal{L}(a) \subseteq M$ with the property that if $S_4 \in \mathcal{L}(a)$ then there exist $X \in \mathcal{L}_1$ and $Y \in \mathcal{L}_2$ with $Y \cap M = S_4$, $X \cap Y = \emptyset$ and $w(X) + w(Y) = a$. Similarly, we enumerate all groups $\mathcal{R}(a) \subseteq M$ with the property that if $S_5 \in \mathcal{R}(b)$ then there exist $X \in \mathcal{R}_1$ and $Y \in \mathcal{R}_2$ with $Y \cap M = S_5$, $X \cap Y = \emptyset$ and $w(X) + w(Y) = b$. In the end we execute a Monte-Carlo algorithm OV that solves the unweighted orthogonal vectors problem that will be described in Theorem 28.

We now analyse the correctness and space usage of this algorithm. The time analysis will be intertwined with the time analysis of Algorithm 2 and is therefore postponed to Subsection 4.5.

Lemma 17. *Algorithm* WeightedOV *is a correct Monte-Carlo algorithm for the Weighted Orthogonal Vectors Problem.*

Proof. If the algorithm outputs true, there exist $A_1 \in \mathcal{L}_1, A_2 \in \mathcal{L}_2, A_3 \in \mathcal{R}_2, A_4 \in \mathcal{R}_1$ such that $w(A_1)+w(A_2)+w(A_3)+w(A_4) = t$.

First, note that by the construction of sets $\mathcal{L}_1, \mathcal{L}_2, \mathcal{R}_1, \mathcal{R}_2$ it has to be that $A_2 \cap A_3 \subseteq M$. Since the OV algorithm checks for disjointness on M we have that $A_2 \cap A_3 \cap M = \emptyset$, hence $A_2 \cap A_3 = \emptyset$. Also, $A_1 \cap A_2 = \emptyset$ because $(X, Y) \in \mathcal{L}(a)$ means $X \cap Y = \emptyset$. Similarly $A_3 \cap A_4 = \emptyset$ because $(X, Y) \in \mathcal{R}(b)$ means that $X \cap Y = \emptyset$. By the construction of the lists $\mathcal{L}_1, \mathcal{L}_2, \mathcal{R}_1, \mathcal{R}_2$ the sets $A_1, \dots, A_4$ are thus mutually disjoint and indeed the instance of Weighted Orthogonal Vectors is a YES-instance.

For the other direction, assume the desired $A_1, \dots, A_4$ quadruple exists. Let $t_L := w(A_1 \cup A_2)$. Then $t_L \in w(\mathcal{L}_1) + w(\mathcal{L}_2)$ and $t - t_L = w(A_3 \cup A_4) \in w(\mathcal{R}_1) + w(\mathcal{R}_2)$. By Lemma 8 inc enumerates $w(\mathcal{L}_1)+w(\mathcal{L}_2)$, and dec enumerates $w(\mathcal{R}_1)+w(\mathcal{R}_2)$ in decreasing order. Therefore, since the loop is a basic linear search routine, it sets a to t_L and b to $t - t_L$ in some iteration: If a is set to t_L before b is set to $t - t_L$, then b is in this iteration larger than $t - t_L$ and it will be decreased in the next iterations until it is set to $t - t_L$. Similarly, if b is set to $t - t_L$ before a is set to t_L, in this iteration a is smaller than t_L and it will be increased in the next iterations until it is set to t_L.

In the iteration with $a = t_L$ and $b = t-t_L$ we have that P^l_a contains the pair $(w(A_1), w(A_2))$ and P^r_b contains the pair $(w(A_4), w(A_3))$. Therefore $\mathcal{L}(a)$ contains $A_2 \cap M = S_4$ and $\mathcal{L}(b)$ contains $A_4 \cap M = S_5$. Since S_4 and S_5 are disjoint a solution will be detected by the OV subroutine with at least constant probability. □

Lemma 18. *Algorithm* WeightedOV *uses at most $O^*(|\mathcal{L}_1| + |\mathcal{L}_2| + |\mathcal{R}_1| + |\mathcal{R}_2| + 2^{|M|})$ space.*

Proof. The datastructures inc and dec use at most $\tilde{O}(|\mathcal{L}_1| + |\mathcal{L}_2| + |\mathcal{R}_1| + |\mathcal{R}_2|)$ space by Lemma 8, and the sets $\mathcal{L}(a)$ and $\mathcal{R}(b)$ are of cardinality at most $2^{|M|}$. The statement follows since, as we will show in Theorem 28, the subroutine $\mathtt{OV}(\mathcal{A}, \mathcal{B})$ uses at most $\tilde{O}(|\mathcal{A}| + |\mathcal{B}| + 2^{|M|})$ space. □

4.3 Correctness of Algorithm 2

We now focus on the correctness of the entire algorithm. First notice that if the algorithm finds a solution on Line 8, it is always correct since it found pairwise disjoint sets A_1, A_2, A_3, A_4 satisfying $w(A_1 \cup A_2 \cup A_3 \cup A_4) = t$. Thus $S := A_1 \cup A_2 \cup A_3 \cup A_4$ is a valid solution. The proof of the reverse implication is less easy and its proof is therefore split in two parts with the help of Lemma 16.

Note that because the partition $[n] = L \uplus M_L \uplus M \uplus M_R \uplus R$ is selected at random, the solution is well-balanced in sets L, M_L, M, M_R, R. The following is a direct consequence of Lemma 10:

Observation 19. *Let S be the solution to the Subset Sum instance with $|S| = \lambda n$. Then, with $\Omega^*(1)$ probability, the following holds: $|S \cap M_L| = \lambda |M_L|$, $|S \cap M| = \lambda |M|$, $|S \cap M_R| = \lambda |M_R|$.*

Now we show that if the above event was successful, the conditions of Lemma 16 apply with good probability:

Lemma 20. *Suppose there exists a solution $S \subseteq [n]$ be such that $w(S) = t$ and $|M_L \cap S| = |M \cap S| = |M_R \cap S| = \lambda\mu n$. Then with probability $\Omega^*(1)$, there exists a partition $S = S_1 \uplus \cdots \uplus S_8$ satisfying all conditions in (4).*

Proof. We select $S_1 = L \cap S$, $S_8 = R \cap S$, and a, b be such that let $a \equiv_{p_L} w(S_1)$ and $b \equiv_{p_R} w(S_8)$. Next we prove that, because the subsets of M generate many distinct sums, the same holds for the solution intersected with M:

Claim 21. *The set $M \cap S$ is an ε'-mixer for some $\varepsilon' \le \varepsilon/\lambda$. Similarly, $M_L \cap S$ is an ε'_L-mixer for $\varepsilon'_L \le \varepsilon_L/\lambda$, and $M_R \cap S$ is an ε'_R-mixer for some $\varepsilon'_R \le \varepsilon_R/\lambda$.*

Proof of Claim 21. Focus on $M \cap S$ (the result for M_L and M_R is analogous). Because M is an ε-mixer, we know that $2^{(1-\varepsilon_L)|M|} \le |w(2^M)| \le |w(2^{M\cap S})||w(2^{M\setminus S})|$. Since $|w(2^{M\setminus S})| \le 2^{(1-\lambda)|M|}$ we have that $|w(2^{M\cap S})| \ge 2^{(\lambda-\varepsilon)|M|} = 2^{(1-\varepsilon/\lambda)|M\cap S|}$. □

Now we know that $Q = M_L \cap S$ is a good mixer. We can use Lemma 7 for $Q = M_L \cap S$ and $p = p_L \cdot p'$, since $|w(2^{|M_L\cap S}|)| \ge$

$2^{(1-\varepsilon_L/\lambda)|M_L \cap S|} = 2^{(\lambda-\varepsilon_L)|M_L|}$. Because x_L was chosen randomly, Lemma 7 guarantees that with $\Omega^*(1)$ probability, there exists $S_2 \subseteq M_L \cap S$, such that $w(S_2) \equiv_{p_L} x_L - a$. Moreover Lemma 7 guarantees that $|S_2| \in [s_0, \lambda\mu n/2]$, where s_0 is the smallest integer such that $\binom{Q}{s_0} \geq w(2^Q)/|Q|$. If we take the logarithm of both sides this is equivalent to

$$\lambda\mu n \cdot h\left(\frac{s_0}{|Q|}\right) \geq \log_2\left(\left|w(2^{(M_L \cap S)})\right|\right) - \frac{\log_2 n}{n}$$
$$\geq (1 - \varepsilon_L/\lambda)\lambda\mu n - \frac{\log_2 n}{n}.$$

Because we have checked all σ_L that satisfy $h(\sigma_L) \geq (1-\varepsilon_L/\lambda) - \frac{\log_2 n}{n}$ the algorithm will eventually guess the correct s_0 (and the same reasoning holds for σ and σ_R). We select $S_3 = (M_L \cap S) \setminus S_2$ with $|S_3| = (1 - \sigma_L)\mu n$.

In a similar manner we can prove that with $\Omega^*(1)$ probability there exists $S_7 \subseteq M_R \cap S$, such that $w(S_7) \equiv_{p_R} (t - x - x_R) - b$ with $|S_7| = \sigma_R \mu n$ and $h(\sigma_R) \geq 1 - \varepsilon_R/\lambda - \frac{\log_2 n}{n}$ (we need to apply Lemma 7 with $Q = M_R \cap S$ and prime p_R). Moreover, this probability only depends on x_R which is independent of all other random variables and events. If this happens, we select $S_6 = (M_R \cap S) \setminus S_7$ with $|S_6| = (1 - \sigma_R)\mu n$.

Conditioned on the existence of $S_1, S_2, S_3, S_6, S_7, S_8$, now we prove there exist S_4 and S_5 with $\Omega^*(1)$ probability. Let $c = w(S_1 \cup S_2 \cup S_3)$ and $d = w(S_6 \cup S_7 \cup S_8)$. We again use Lemma 7, but this time with $Q = M \cap S$ and $p = p_R \cdot p'$. It assures that with high probability there exist $S_4 \subseteq M \cap S$, with $w(S_4) \equiv_{p_L} x - c$ and $|S_4| = \sigma\mu n$ with $h(\sigma) \geq 1 - \varepsilon_L/\lambda - \frac{\log_2 n}{n}$. And indeed, again this probability only depends on x_R which is independent of all other random variables and events. If this event happens, we select $S_5 = (M \cap S) \setminus S_4$.

Now we use the fact that p_R divides p_L: If $x \equiv_{p_L} a$ then $x \equiv_{p_R} a$ because $(x - a) = k \cdot p' \cdot p_R$ for some $k \in \mathbb{Z}$. Hence $w(S_5) + d \equiv_{p_R} w(S) - x$, which means that $w(S_5 \cup S_6 \cup S_7 \cup S_8) \equiv_{p_R} t - x$. Moreover it holds that $|S_5| = (1 - \sigma)\mu n$, thus S_5 also satisfies the desired conditions. To conclude observe that the constructed sets $S_1, \ldots, S_8$ are disjoint. □

Finally, we prove the Lemma 16. Namely, we show that the existence of the tuple $(S_1, \ldots, S_8)$ implies that a solution is detected.

Proof of Lemma 16. By the construction of $\mathcal{L}_1, \mathcal{L}_2, \mathcal{R}_1, \mathcal{R}_2$ and the assumed properties of the lemma, we have that $A_1 := S_1 \cup S_2 \in \mathcal{L}_1$, $A_2 := S_3 \cup S_4 \in \mathcal{L}_2$, $A_3 := S_5 \cup S_6 \in \mathcal{R}_2$, and $A_4 := S_7 \cup S_8 \in \mathcal{R}_1$. Since the sets $S_1, \ldots, S_8$ are pairwise disjoint and satisfy $\sum_{i=1}^{8} w(S_i) = t$, the sets $A_1, \ldots, A_4$ certify that our instance of Weighted Orthogonal Vectors instance is true. □

The correctness of Algorithm 2 directly follows by combining Lemma 20 and Lemma 16.

4.4 Space Usage of Algorithm 2

The bulk of the analysis of the space usage consists of computing the expected sizes of the lists $\mathcal{L}_1, \mathcal{L}_2, \mathcal{R}_1, \mathcal{R}_2$. This requires us to look closely into the setting of the parameters.

Useful bounds on parameters. Recall, that we defined the following constants $\lambda_0 := 0.495$ and $\varepsilon_0 := 0.00002$. Then, we assumed that $\varepsilon, \varepsilon_L, \varepsilon_R \leq \varepsilon_0$ and $\lambda \in [\lambda_0, 0.5]$. Moreover, we have chosen $\sigma, \sigma_L, \sigma_R$, such that:

$$0.99995 < 1 - \varepsilon_0/\lambda_0 - \frac{\log_2 n}{n} \leq h(\sigma), h(\sigma_L), h(\sigma_R)$$

Which means that (for ε_0 and λ_0 and large enough n):

$$\sigma, \sigma_L, \sigma_R \in [0.495, 0.505]. \tag{10}$$

because $h(0.495) = h(0.505) \approx 0.999928$. Next, observe that

$$h(\sigma\lambda), h((1 - \sigma)\lambda) \leq h(1/4) + 0.004. \tag{11}$$

because the entropy function is increasing in $[0, 0.5]$ and $h(0.5 \cdot 0.505) - h(1/4) < 0.004$. For the next inequality, recall that $\beta(\sigma, \lambda) = h(\sigma\lambda) - h((1 - \sigma)\lambda)$.

$$-0.012 \leq \beta(\sigma, \lambda) \leq 0.012 \tag{12}$$

because $|\beta| < h(0.505 \cdot 0.5) - h(0.495 \cdot \lambda_0) < 0.012$.

Bounds on the list sizes.

Claim 22. $\mathbb{E}[|\mathcal{L}_1|] \leq O^*\left(2^{(1/2-\mu(3/2+\lambda-h(1/4)-0.02))n}\right)$.

Proof. Let W_L be the number of possible different elements from $\mathcal{L}_1$. It is

$$W_L := 2^{|L|}\binom{\mu n}{\lambda\sigma_L\mu n}.$$

The expected size of $\mathcal{L}_1$ over the random choices of x_L is $\leq \frac{W_L}{p_L}$. If we plug in the definition of $|L|$, we have:

$$(\log_2(\mathbb{E}[|\mathcal{L}_1|])/n) \leq 1/2 - \mu(3/2 + \lambda - h(\lambda\sigma_L)) + \mu(\varepsilon_L - \beta/2).$$

By (11) we have that $h(\sigma_L\lambda) \leq h(1/4) + 0.004$. By (12) we have that $|\beta| \leq 0.012$ and $\varepsilon_L < 0.01$. Hence:

$$(\log_2(\mathbb{E}[|\mathcal{L}_1|])/n) \leq 1/2 - \mu(3/2 + \lambda - h(1/4)) + 0.02 \cdot \mu. \quad \square$$

By symmetry[8] the same bound holds for $\mathbb{E}[|\mathcal{R}_1|]$.

Claim 23. $\mathbb{E}[|\mathcal{R}_1|] \leq O^*\left(2^{(1/2-\mu(3/2+\lambda-h(1/4)-0.02))n}\right)$.

Next we bound $|\mathcal{L}_2|$ and $|\mathcal{R}_2|$:

Claim 24. $\mathbb{E}[|\mathcal{L}_2|] \leq O^*(2^{\mu n(2h(1/4)-\lambda)+0.02\mu n})$.

Proof. Let W_L be the number of possibilities of selecting S. It is

$$W_L := \binom{\mu n}{\sigma\lambda\mu n}\binom{\mu n}{(1 - \sigma_L)\lambda\mu n}$$

The expected size of $|\mathcal{L}_2|$ over the random choices of x_L and p_L is $\mathbb{E}[|\mathcal{L}_2|] \leq \frac{W_L}{p_L}$. Hence,

$$(\log_2(\mathbb{E}[|\mathcal{L}_2|]))/n \leq \mu(h(\lambda\sigma) + h(\lambda(1 - \sigma_L)) - \lambda + \varepsilon_L).$$

We use Inequality 11 and have $h((1 - \sigma)\lambda), h(\sigma\lambda) \leq h(1/4) + 0.004$. Hence we can roughly bound:

$$(\log_2(\mathbb{E}[|\mathcal{L}_2|]))/n \leq \mu(2h(1/4) - \lambda) + 0.02 \cdot \mu. \quad \square$$

By symmetry, the same bound holds for $|\mathcal{R}_2|$:

Claim 25. $\mathbb{E}[|\mathcal{R}_2|] \leq O^*(2^{\mu n(2h(1/4)-\lambda)+0.02\mu n})$.

[8] The only difference being that β shows up positively rather than negatively, but this does not matter since we bound its absolute value.

As mentioned in Subsection 4.1, the subroutine `WeightedOV` uses $O^*(|\mathcal{L}_1|+|\mathcal{L}_2|+|\mathcal{R}_1|+|\mathcal{R}_2|+2^{|M|})$ space. By the above claims, we see that this is at most

$$O^*\left(2^{(1/2-\mu(3/2+\lambda-h(1/4)))n+0.02\mu n}+2^{\mu n(2h(1/4)-\lambda)+0.02\mu n}+2^{\mu n}\right)$$

as promised.

Remark 26. *The constant* 0.02 *is based on our choice of* ε_0 *and* λ_0*. When* $\varepsilon \to 0$ *and* $\lambda_0 \to 1/2$ *it goes to* 0*. With more complicated inequalities and a tighter choice of parameters we were able to get* $O^*(2^{0.249936n})$ *space usage. We decided to skip the details for the simplicity of the presentation.*

Remark 27. *In this section we showed that expected sizes of* $\mathcal{L}_1$,$\mathcal{L}_2$, $\mathcal{R}_1$,$\mathcal{R}_2$ *are bounded by* $2^{(0.25-\delta)n}$ *for some constant* $\delta > 0$*. With a standard Markov's inequality and union bound one can show that with* $\Omega^*(1)$ *probability it holds that sizes of* $\mathcal{L}_1, \mathcal{L}_2, \mathcal{R}_1, \mathcal{R}_2$ *are bounded by* $2^{(0.25-\delta)n}$ *for some constant* $\delta > 0$.

4.5 Runtime of Algorithm 2

Now, we prove that the runtime of Algorithm 2 is $O^*(2^{n/2})$. By Lemma 8, the total runtime of all queries to inc.next() is $O^*(|\mathcal{L}_1|\,|\mathcal{L}_2|)$, and the total runtime of all the queries to dec.next() is $O^*(|\mathcal{R}_1||\mathcal{R}_2|)$. This is upper bounded by $O^*(2^{n/2})$ by the analysis of Subsection 4.4.

The main bottleneck of the algorithm comes from all the calls to OV subroutine. To facilitate the analysis, we define sets $\mathcal{A}, \mathcal{B}$ that represent the total input to the OV subroutine: For every $a \in \mathbb{N}$, such that $a \equiv_{p_L} x$ and each $X \in \mathcal{L}(a)$, add the pair $(X \cap M, a)$ to $\mathcal{A}$ (without repetitions). Similarly, for each $Y \in \mathcal{R}(t-a)$, add the pair $(Y \cap M, t-a)$ to $\mathcal{B}$. Hence the total input for OV generated by is:

$$\mathcal{A} := \left\{(X,a) : X \in \binom{M}{\sigma\lambda\mu n},\ a - w(X) \in w(2^{L\cup M_L}),\ a \equiv_{p_L} x\right\},$$

$$\mathcal{B} := \left\{(Y,b) : Y \in \binom{M}{(1-\sigma)\lambda\mu n},\ b - w(Y) \in w(2^{R\cup M_R}),\ b \equiv_{p_R} t-x\right\}.$$

Now, let us calculate the expected size of $\mathcal{A}$. The number of possibilities of selecting possible elements in $\mathcal{A}$ is the number of possibilities of selecting X from M and a from $w(2^{L\cup M_L})$. Since the probability that $a \equiv_{p_L} x$ is $1/p_L$, we obtain

$$\mathbb{E}\left[|\mathcal{A}|\right] \le \binom{M}{\sigma\lambda\mu n}|w(2^{L\cup M_L})|/p_L.$$

Similarly, the probability that $b \equiv_{p_R} t-x$ is $1/p_R$. To see this recall that x is chosen uniformly at random from Z_{p_L}, but since p_L is a multiple of p_R, integer $x \mod p_R$ is also uniformly distributed in $\mathbb{Z}_{p_R}$.

$$\mathbb{E}\left[|\mathcal{B}|\right] \le \binom{M}{(1-\sigma)\lambda\mu n}|w(2^{R\cup M_R})|/p_R.$$

Recall that M_L is an ε_L-mixer, hence $|w(2^{L\cup M_L})| \le |w(2^{|L|})|\,2^{(1-\varepsilon_L)\mu n}$, and similarly M_R is an ε_R-mixer. Hence:

$$\begin{aligned}\log_2\left(\mathbb{E}\left[|\mathcal{A}|\right]\right) &\le |L| + (1-\varepsilon_L)\mu n + h(\sigma\lambda)\mu n - (\lambda-\varepsilon_L)\mu n\\ &= \left(\frac{1-3\mu-\beta\mu}{2} + \mu - \lambda\mu + h(\sigma\lambda)\mu\right)n,\\ &= \left(\frac{1}{2} - \mu\left(\frac{1}{2} + \lambda + \beta/2 - h(\sigma\lambda)\right)\right)n,\end{aligned}$$

and similarly:

$$\begin{aligned}\log_2\left(\mathbb{E}\left[|\mathcal{B}|\right]\right) &\le |R| + (1-\varepsilon_R)\mu n + h((1-\sigma)\lambda)\mu n - (\lambda-\varepsilon_R)\mu n\\ &= \left(\frac{1-3\mu+\beta\mu}{2} + \mu - \lambda\mu + h((1-\sigma)\lambda)\mu\right)n\\ &= \left(\frac{1}{2} - \mu\left(\frac{1}{2} + \lambda - \beta/2 - h((1-\sigma)\lambda)\right)\right)n.\end{aligned}$$

Now it becomes clear that we have chosen the balancing parameter β in the sizes $|L|, |R|$ to match the sizes of $\mathcal{A}, \mathcal{B}$: Observe that

$$\beta/2 - h(\sigma\lambda) = -\frac{h(\sigma\lambda) + h((1-\sigma)\lambda)}{2} = -\beta/2 - h((1-\sigma)\lambda),$$

and thus we obtain that $\log_2\left(\mathbb{E}\left[|\mathcal{A}|\right]\right)$ and $\log_2\left(\mathbb{E}\left[|\mathcal{B}|\right]\right)$ are less or equal to

$$\left(\frac{1}{2} - \mu\left(\frac{1}{2} + \lambda - \frac{h(\sigma\lambda) + h((1-\sigma)\lambda)}{2}\right)\right)n.$$

By the concavity of binary entropy function (see (3)), we know that $h(\sigma\lambda) + h((1-\sigma)\lambda) \le 2h(\lambda/2)$. Hence:

$$\mathbb{E}\left[|\mathcal{A}|\right], \mathbb{E}\left[|\mathcal{B}|\right] \le O^*(2^{n/2-\mu n(1/2+\lambda-h(\lambda/2))}). \tag{13}$$

The OV subroutine (see Theorem 28) takes $\mathcal{A}$ and $\mathcal{B}$ as an input with dimension $d = \mu n$. Note that the condition $\lambda \in [0.4, 0.5]$ in Theorem 28 is satisfied by the assumption in the Lemma 15 and $\sigma \in [0.4, 0.6]$ is satisfied because for our choice of parameters $\sigma \in [0.495, 0.505]$ (see (10)). Since the run time of the OV subroutine is linear in the input size, all calls to the OV algorithms jointly take the following total run time:

$$O^*\left((|\mathcal{A}| + |\mathcal{B}|)\, 2^{\mu n(1/2+\lambda-h(\lambda/2))}\right).$$

Thus the algorithm runs in $O^*(2^{n/2})$ time by (13).

5 ORTHOGONAL VECTORS VIA REPRESENTATIVE SETS

In this section we present and discuss our algorithm for Orthogonal Vectors. As discussed in the introduction it should be noted that the proof strategy is similar to the one from [21] (which is heavily inspired on Bollobás's Theorem [14]), but we obtain improvements that are crucial for the main result of this paper. We compare our methods with existing literature at the end of this section.

Theorem 28 (OV-algorithm, Generalization of Theorem 2). *For any* $\lambda \in [0.4, 0.5]$ *and* $\sigma \in [0.4, 0.6]$*, there is a Monte-Carlo algorithm that is given* $\mathcal{A} \subseteq \binom{d}{\sigma\lambda d}$ *and* $\mathcal{B} \subseteq \binom{d}{(1-\sigma)\lambda d}$*, detects if there exist* $A \in \mathcal{A}$ *and* $B \in \mathcal{B}$ *with* $A \cap B = \emptyset$ *in time*

$$\tilde{O}\left((|\mathcal{A}| + |\mathcal{B}|)\, 2^{d(1/2+\lambda-h(\lambda/2))}\right)$$

and space $\tilde{O}(|\mathcal{A}| + |\mathcal{B}| + 2^d)$.

We can assume that $\lambda \leq 0.5$ by a subset complementation trick. The bound $\sigma \in [0.4, 0.6]$ is an artifact of technical methods we used in the proof of Lemma 9. In the proof of this lemma the parameters λ and σ lost their meaning from Section 4. Hence, to simplify, we let $p := \sigma\lambda n$ and $q := (1-\sigma)\lambda n$, and let $\mathcal{A} \subseteq \binom{[d]}{p}$ and $\mathcal{B} \subseteq \binom{[d]}{q}$. We use the following standard definitions from communication complexity (see for example [36]):

Definition 29 ((p, q, d)-Disjointness Matrix)**.** *For integers p, q, d the Disjointness matrix $\mathtt{Disj}_{p,q,d}$ has its rows indexed by $\binom{[d]}{p}$ and its columns indexed by $\binom{[d]}{q}$. For $A \in \binom{[d]}{p}$ and $B \in \binom{[d]}{q}$ we define*

$$\mathtt{Disj}_{p,q,d}[A, B] = \begin{cases} 1 & \text{if } A \cap B = \emptyset, \\ 0 & \text{otherwise.} \end{cases}$$

Definition 30 (Monochromatic Rectangle, 1-Cover)**.** *A* monochromatic rectangle *of a matrix M is subset X of rows and subset Y of the columns such that $M[i, j] = M[i', j']$ for every $i, i' \in X$ and $j, j' \in Y$. A family of monochromatic rectangles $\mathcal{M} = (X_1, Y_1), \dots, (X_z, Y_z)$ is called a* 1-cover *if for every i, j such that $M[i, j] = 1$, there exists $k \in [z]$, such that $i \in X_k$ and $j \in Y_k$.*

A natural goal in the field of communication complexity is to find 'good' 1-covers. The natural parameter that quantifies such 'goodness' is z (intuitively the smaller z the better a 1-cover we have). The parameter z is sometimes called the Boolean rank[9] and it is known to be equal to $2^{\mathsf{nc}(M)}$ where $\mathsf{nc}(M)$ is the 'non-deterministic communication complexity' of M (see e.g. [36]).

Such 1-covers of the Disjointness matrix can be used in algorithms for the Orthogonal Vectors problem: An orthogonal pair is a 1 in the submatrix of the Disjointness induced by the rows and columns from the families $\mathcal{A}$ and $\mathcal{B}$, and we can search for such a 1 via searching for the associated monochromatic rectangle that covers it (see Lemma 32 for a related approach). For the case that $p = q$, it is well known that $\mathtt{Disj}_{p,q,d}$ admits a 1-cover with $O(2^{2p} p \ln d)$ rectangles [36, Claim 1.37]. When applied naïvely, this 1-cover would imply an $\tilde{O}((|\mathcal{A}| + |\mathcal{B}|)2^{d/2})$ time algorithm for the setting of Theorem 2 with $p = q = d/4$.

In order to get a faster algorithm we introduce the following new parameter of a 1-cover:

Definition 31 (Sparsity)**.** *The* sparsity *of a 1-cover $\mathcal{M} = (X_1, Y_1), \dots, (X_z, Y_z)$ of an $n \times m$ matrix is defined as $\sum_i |X_i|/n + \sum_i |Y_i|/m$.*

A 1-cover of sparsity Ψ of a matrix can be understood as a factorization of $M = L \cdot R$ over the Boolean semi-ring such that the average number of 1's in a row L plus the average number of 1's in a column of R is at most Ψ. Our notion of sparsity is related to the degree of the data structure called *n-p-q-separating collection* [21]. For a further discussion about sparse factorizations see [32, Section 5.1])

We present the algorithmic usefulness of the notion of the sparsity of 1-cover with the following statement.

Lemma 32 (Orthogonal Vectors Parameterized by the Sparsity)**.** *For any constant integer c and integers p, q, d such that c divides p, q, d, there is an algorithm that takes as an input a 1-cover $\mathcal{M}$ of $\mathtt{Disj}_{p/c,q/c,d/c}$ of sparsity Ψ and two set families $\mathcal{A} \subseteq \binom{[d]}{p}$, $\mathcal{B} \subseteq \binom{[d]}{q}$ with the following properties: It outputs a pair $A \in \mathcal{A}$ and $B \in \mathcal{B}$ such that $A \cap B = \emptyset$ with constant non-zero probability if such a pair exists. Moreover, it uses $\tilde{O}((|\mathcal{A}| + |\mathcal{B}|)\Psi^c + 2^{2d/c})$ time and $\tilde{O}(2^{2d/c} + z^c)$ space, where z is the number of rectangles of $\mathcal{M}$.*

Proof. Denote the 1-cover to be $\mathcal{M} = (X_1, Y_1), \dots, (X_z, Y_z)$. Observe, that if $A \cap B = \emptyset$ then it suffices to find $\ell \in [z]$ such that $A \in X_\ell$ and $B \in Y_\ell$ since $\mathcal{M}$ forms a 1-cover. In the bird's eye view, the algorithm will find such an ℓ. We need to make sure that the space usage of our algorithm is low. We will use parameter c to achieve that (it is instructive for a reader to assume $c = 1$). In the full version of this paper [35] we present a pseudocode with an overview of the proof.

First, randomly partition $[d]$ into blocks $U_1, \dots, U_c$ with $|U_d| = d/c$. By Lemma 10, if we repeat the algorithm $d^{O(c)}$ times with probably at least $1/d^{O(c)}$ this partition is good, i.e., for some orthogonal pair A, B it holds that $|A \cap U_i| = p/c$ and $|B \cap U_i| = q/c$.

Next, we map the given factorization $X_1, \dots, X_z, Y_1, \dots, Y_z$ of $\mathtt{Disj}_{p/c,q/c,d/c}$, to the set U_i by unifying U_i with $[d]$ a uniformly random permutation.

Now we present a processing step of the algorithm. For every $i \in [c]$ we create and store two lists L and R. The purpose of these lists is to give every element in A and B fast access to corresponding rectangles from the 1-cover that contain it (i.e., given A we need to find all X_i, such that $A \in X_i$ in $\tilde{O}(z)$ time). Specifically, for every $i \in [c]$ construct:

$$\text{For every } Q \in \binom{[d/c]}{p/c} \text{ construct } L_i(Q) := \{j \in [z] \ : \ Q \in X_j\}.$$

And similarly for all $i \in [c]$:

$$\text{For every } Q \in \binom{[d/c]}{q/c} \text{ construct } R_i(Q) := \{j \in [z] \ : \ B \in Y_j\}.$$

Because $\binom{d/c}{p/c} \leq 2^{d/c}$ we can construct and store all $L_i(Q)$ and $R_i(Q)$ in $\tilde{O}(2^{2d/c} + 2^{d/c}z)$ time and space. Additionally, initialize a table $T[i_1, \dots, i_c] := \mathtt{false}$ for every $i_1, \dots, i_c \in [c]$. This table will store which sets X_i, Y_i have been seen by elements in $\mathcal{A}$. Observe that so far we did not look at the input $\mathcal{A}$ and $\mathcal{B}$; we just preprocessed the 1-cover, so the next steps can be computed efficiently.

Now iterate over every element $A \in \mathcal{A}$ and check if we can afford to process it, i.e., if $|L_1(A \cap U_1)| \cdots |L_c(A \cap U_c)| > (4c\Psi)^c$ we simply ignore it (later we will prove that for a disjoint pair A and B this situation happens with low probability). If indeed we can afford it, then we mark it in table T: For every $(i_1, \dots, i_c) \in L_1(A \cap U_1) \times \ldots \times L_c(A \cap U_c)$ we mark $T(i_1, \dots, i_c)$ to be $\mathtt{true}$. Clearly this step takes $\tilde{O}(|\mathcal{A}|\Psi^c)$ time.

Next, we treat $\mathcal{B}$ in a similar way: We iterate over every element $B \in \mathcal{B}$ and check if $|R_1(B \cap U_1)| \cdots |R_c(B \cap U_c)| \leq (4c\Psi)^c$. If so, we iterate over every $(i_1, \dots, i_c) \in R_1(B \cap U_1) \times \ldots \times R_c(B \cap U_c)$ and check if $T(i_1, \dots, i_c) = \mathtt{true}$. If this happens, then it means there exists $A \in \mathcal{A}$ that is orthogonal to the current B and we can return $\mathtt{true}$. If this never happens, we return $\mathtt{false}$. Clearly, the total running time of the algorithm is $\tilde{O}((|\mathcal{A}| + |\mathcal{B}|)\Psi^c)$ and extra amount of working memory is $\tilde{O}(2^{2d/c} + z^c)$. Hence we focus on correctness.

[9] The name 'Boolean rank' is used because a 1-cover of M with z rectangles is equivalent to a factorization $M = L \cdot R$ over the Boolean semi-ring of rank z.

Note that if `true` is returned, indeed there must exist disjoint $A \in \mathcal{A}$ and $B \in \mathcal{B}$ because $\mathcal{M}$ is 1-cover. For the other direction, suppose that there exist orthogonal $A \in \mathcal{A}$ and $B \in \mathcal{B}$. As mentioned this implies by Lemma 10 that with $1/d^c$ we have that for each i it holds that $|A \cap U_i| = p/c$ and $|B \cap U_i| = p/c$. Because we unified $[d]$ with U_i with a random permutation, $\mathbb{E}[|L_i(A \cap U_i)|], \mathbb{E}[|R_i(A \cap U_i)|] = \Psi$, and by Markov's inequality and a union bound there will be no i with $|L_i(A \cap U_i)| + |R_i(B \cap U_i)| \geq 4c\Psi$, and therefore $|L_1(A \cap U_1)| \cdots |L_c(A \cap U_c)| \leq (4c\Psi)^c$ and $|R_1(B \cap U_1)| \cdots |R_c(B \cap U_c)| \leq (4c\Psi)^c$. If this happens, the orthogonal pair will be detected since $(X_1, Y_1), \dots, (X_z, Y_z)$ is a 1-cover. □

Lemma 33 (Construction of 1-cover with small sparsity)**.** *Let p, q and d be integers such that $p \leq q$ and $p + q \leq d/2$. There is a randomized algorithm that in $O(2^d)$ time and space, constructs $X_1, \dots, X_z \subseteq \binom{[d]}{p}$ and $Y_1, \dots, Y_z \subseteq \binom{[d]}{p}$, where z is at most 2^d.*

All pairs of sets $(X_1, Y_1), \dots, (X_z, Y_z)$ form monochromatic rectangles in $\mathtt{Disj}_{p,q,d}$ and with probability at least $3/4$, it holds that $(X_1, Y_1) \dots, (X_z, Y_z)$ is a 1-cover of $\mathtt{Disj}_{p,q,d}$ with sparsity

$$d^{O(1)} \cdot 2^{d/2+p+q-d\cdot h\left(\frac{p+q}{2d}\right)}.$$

Proof. Let $l = p + q$ and let $A \in \mathcal{A}, B \in \mathcal{B}$ be an orthogonal pair. Let x be some parameter that we will determine later (think about $x \approx d/2$). Note that

$$\left|\left\{S \in \binom{[d]}{x} : A \subseteq S \text{ and } S \cap B = \emptyset\right\}\right| = \binom{d-l}{x-p}.$$

Let $\mathcal{S} := \{S_1, \dots, S_z\} \subseteq \binom{[d]}{x}$ be obtained by including each set from $\binom{[d]}{x}$ with probability $2d\binom{d-l}{x-p}^{-1}$ (assuming $x > p + \Omega(1)$, this probability is indeed in the interval $[0, 1]$).

Thus, if A and B are disjoint sets, with good probability there is a certificate set $S \in \mathcal{S}$, such that $A \subseteq S$ and $S \cap B = \emptyset$. More formally:

$$\mathbb{P}\left[\nexists S \in \mathcal{S} : A \subseteq S \text{ and } S \cap B = \emptyset \;\middle|\; A \cap B = \emptyset\right] = \tag{14}$$

$$= \left(1 - 2d\binom{d-l}{x-p}^{-1}\right)^{\binom{d-l}{x-p}} \leq \exp(-2d),$$

(where the last inequality is due to the standard inequality $1 + \alpha \leq \exp(\alpha)$). Now we define a 1-cover based on the family $\mathcal{S}$:

$$\text{For every } i \in [z]: \quad X_i := \binom{S_i}{p} \text{ and } Y_i := \binom{[d] \setminus S_i}{q}.$$

First let us prove that with good probability $X_1, Y_1 \dots, X_z, Y_z$ is 1-cover. There are at most 3^d disjoint pairs A, B. Hence by Equation 14 and the union bound on all disjoint pairs A, B, we have that $(X_1, Y_1), \dots, (X_z, Y_z)$ is a 1-cover with probability at least $3/4$.

Next, we bound the sparsity of $(X_1, Y_1), \dots, (X_z, Y_z)$. By Markov's inequality, $z \leq 4d\binom{d}{x}\binom{d-l}{x-p}^{-1}$ with probability at least $1/2$. Hence with probability at least $1/2$ our 1-cover has sparsity at most:

$$\begin{aligned} &4d\binom{d}{x}\binom{d-l}{x-p}^{-1}\left(|X_i|/\binom{d}{p} + |Y_i|/\binom{d}{q}\right) = \\ &4d\binom{d}{x}\binom{d-l}{x-p}^{-1}\left(\binom{x}{p}\binom{d}{p}^{-1} + \binom{d-x}{q}\binom{d}{q}^{-1}\right) = \\ &4d\left(\binom{d-p}{x-p} + \binom{d-q}{x}\right)\binom{d-l}{x-p}^{-1}, \end{aligned} \tag{15}$$

where the second equality follows from using $\binom{a}{b+c}\binom{b+c}{c} = \binom{a}{b,c} = \binom{a}{c}\binom{a-c}{b}$ twice.

Next, we use Lemma 9 with: $d = n$, $p = \sigma\lambda n$, $q = (1-\sigma)\lambda n$ and $p + q = \lambda n$. Note that we assumed that $\sigma \in [0.4, 0.6]$ and $\lambda \in [0.4, 0.5]$ hence conditions for Lemma 9 are satisfied. We obtain that for the choice of $x := d(1/2 + (\sigma - 1/2)(\log_2(3)/2) + (1/2 - \sigma)(1/2 - \lambda))$ expression (15) is bounded from above with

$$d^{O(1)} \cdot 2^{d/2+p+q-d\cdot h(\frac{p+q}{2d})},$$

as required. □

Now the main statement of this section follows by a straightforward combination of the previous lemmas:

Proof of Theorem 28. Let $p := \sigma\lambda d$ and $q := (1 - \sigma)\lambda d$. Set $c = 20$ and assume that integers p, q, d are multiples of c (by padding the instance if needed).

Next, use Lemma 33 with d/c, p/c and q/c to construct a 1-cover $\mathcal{M}$ of sparsity

$$\Psi = d^{O(1)} \cdot 2^{\frac{d/2+p+q-dh((p+q)/(2d))}{c}},$$

with good probability. Subsequently, apply Lemma 32 with this 1-cover $\mathcal{M}$ to detect a disjoint pair $A \in \mathcal{A}$ and $B \in \mathcal{B}$ with constant probability. Note that the runtime is:

$$\tilde{O}\left((|\mathcal{A}| + |\mathcal{B}|)\,(4c\Psi)^c + 2^{2d/c}\right).$$

This is equal to

$$\tilde{O}\left((|\mathcal{A}| + |\mathcal{B}|)\,2^{d/2+p+q-dh((p+q)/2d)}\right).$$

Hence, the running time is $\tilde{O}\left((|\mathcal{A}| + |\mathcal{B}|)\,2^{d(1/2+\lambda-h(\lambda/2))}\right)$. The main bottleneck in the space usage comes from the z^c factor in Lemma 32 which gives the 2^d factor. □

Lower bound on sparsity. One might be tempted to try to get even better bounds on the sparsity of the disjointness matrix. Here we show that the sparsity bound from Lemma 33 is essentially optimal with a fairly straightforward counting argument. It means that new techniques would have to be developed to improve an algorithm for Orthogonal Vectors in the worst case $\sigma = 1/2$ and $\lambda = 1/2$, and in consequence improve the meet-in-middle algorithm for Subset Sum.

Theorem 34. *Any 1-cover of $\mathtt{Disj}_{d/4,d/4,d/2}$ has sparsity at least $\Omega^*\left(2^d/\binom{d}{d/4}\right)$.*

We defer the proof of this statement to the full version of the paper [35].

Relation of techniques in this section with existing methods. The idea for constructing the 1-cover is relatively standard in communication complexity (see e.g., the aforementioned [36, Claim 1.37]). It was also used in some proofs of Bollobás's Theorem [14]. The idea of randomly partitioning the universe to get a structured 1-cover is very similar to the derandomization of the color-coding approach from [5].

Both ideas were also used by [21]. They also start with a probabilistic construction (c.f., [21, Lemma 4.5]) on a small universe that is repeatedly applied, and use it to set up a data structure of 'n-p-q-separating collections' that is similar to our lists.[10] The small but crucial difference, however is that (in our language) they obtain a monochromatic rectangle by sampling a random set $S \subseteq [d]$ (in contrast to our random sampling $S \in \binom{[d]}{d/4}$ in the case $p = q = d/4$), and in the case $p = q = d/4$ this would lead to sparsity $2^{3d/4}/2^{d/4} \gg 2^d/\binom{d}{d/4}$.

ACKNOWLEDGEMENT

We thank the anonymous reviewers for their insightful comments and suggestions. The first author would like to thank Per Austrin, Nikhil Bansal, Petteri Kaski, Mikko Koivisto for several inspiring discussions about reductions from Subset Sum to Orthogonal Vectors. The second author would like to thank Marcin Mucha and Jakub Pawlewicz for useful discussions.

REFERENCES

[1] Amir Abboud. 2020. personal communication.

[2] Amir Abboud, Karl Bringmann, Danny Hermelin, and Dvir Shabtay. 2019. SETH-Based Lower Bounds for Subset Sum and Bicriteria Path. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019*. 41–57.

[3] Amir Abboud and Kevin Lewi. 2013. Exact Weight Subgraphs and the k-Sum Conjecture. In *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013 (Lecture Notes in Computer Science, Vol. 7965)*. Springer, 1–12.

[4] Amir Abboud, Richard Ryan Williams, and Huacheng Yu. 2015. More Applications of the Polynomial Method to Algorithm Design. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015*. SIAM, 218–230.

[5] Noga Alon, Raphael Yuster, and Uri Zwick. 1995. Color-Coding. *J. ACM* 42, 4 (1995), 844–856.

[6] Per Austrin, Petteri Kaski, Mikko Koivisto, and Jussi Määttä. 2013. Space-Time Tradeoffs for Subset Sum: An Improved Worst Case Algorithm. In *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013*. 45–56.

[7] Per Austrin, Petteri Kaski, Mikko Koivisto, and Jesper Nederlof. 2015. Subset Sum in the Absence of Concentration. In *32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015*. 48–61.

[8] Per Austrin, Petteri Kaski, Mikko Koivisto, and Jesper Nederlof. 2016. Dense Subset Sum May Be the Hardest. In *33rd Symposium on Theoretical Aspects of Computer Science, STACS 2016*. 13:1–13:14.

[9] Nikhil Bansal, Shashwat Garg, Jesper Nederlof, and Nikhil Vyas. 2018. Faster Space-Efficient Algorithms for Subset Sum, k-Sum, and Related Problems. *SIAM J. Comput.* 47, 5 (2018), 1755–1777.

[10] Anja Becker, Jean-Sébastien Coron, and Antoine Joux. 2011. Improved Generic Algorithms for Hard Knapsacks. In *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques. Proceedings*. 364–385.

[11] Andreas Björklund. 2014. Determinant Sums for Undirected Hamiltonicity. *SIAM J. Comput.* 43, 1 (2014), 280–299.

[12] Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. 2009. Counting Paths and Packings in Halves. In *Algorithms - ESA 2009, 17th Annual European Symposium. Proceedings*, Amos Fiat and Peter Sanders (Eds.).

[13] Andreas Björklund, Thore Husfeldt, and Mikko Koivisto. 2009. Set Partitioning via Inclusion-Exclusion. *SIAM J. Comput.* 39, 2 (2009), 546–563.

[14] Béla Bollobás. 1965. On generalized graphs. *Acta Mathematica Academiae Scientiarum Hungarica* 16, 3-4 (1965), 447–452.

[15] Karl Bringmann. 2017. A Near-linear Pseudopolynomial Time Algorithm for Subset Sum. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017*. 1073–1084.

[16] Karl Bringmann. 2020. personal communication.

[17] Timothy M. Chan and Ryan Williams. 2016. Deterministic APSP, Orthogonal Vectors, and More: Quickly Derandomizing Razborov-Smolensky. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016*. SIAM, 1246–1255.

[18] Lijie Chen and Ryan Williams. 2019. An Equivalence Class for Orthogonal Vectors. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019*.

[19] Marek Cygan, Holger Dell, Daniel Lokshtanov, Dániel Marx, Jesper Nederlof, Yoshio Okamoto, Ramamohan Paturi, Saket Saurabh, and Magnus Wahlström. 2016. On Problems as Hard as CNF-SAT. *ACM Trans. Algorithms* 12, 3 (2016), 41:1–41:24.

[20] Itai Dinur, Orr Dunkelman, Nathan Keller, and Adi Shamir. 2012. Efficient Dissection of Composite Problems, with Applications to Cryptanalysis, Knapsacks, and Combinatorial Search Problems. In *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference. Proceedings*.

[21] Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. 2016. Efficient Computation of Representative Families with Applications in Parameterized and Exact Algorithms. *J. ACM* 63, 4 (2016), 29:1–29:60.

[22] Jiawei Gao, Russell Impagliazzo, Antonina Kolokolova, and Ryan Williams. 2019. Completeness for First-order Properties on Sparse Structures with Algorithmic Applications. *ACM Trans. Algorithms* 15, 2 (2019), 23:1–23:35.

[23] Ellis Horowitz and Sartaj Sahni. 1974. Computing Partitions with Applications to the Knapsack Problem. *J. ACM* 21, 2 (1974), 277–292.

[24] Nick Howgrave-Graham and Antoine Joux. 2010. New Generic Algorithms for Hard Knapsacks. In *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques. Proceedings*. 235–256.

[25] Ce Jin and Hongxun Wu. 2019. A Simple Near-Linear Pseudopolynomial Time Randomized Algorithm for Subset Sum. In *2nd Symposium on Simplicity in Algorithms, SOSA@SODA 2019*. 17:1–17:6.

[26] Konstantinos Koiliaris and Chao Xu. 2017. A Faster Pseudopolynomial Time Algorithm for Subset Sum. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017*.

[27] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. 2011. Lower bounds based on the exponential time hypothesis. *Bulletin of the European Association for Theoretical Computer Science EATCS* 105 (01 2011).

[28] Daniel Lokshtanov and Jesper Nederlof. 2010. Saving Space by Algebraization. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010*. 321–330.

[29] Burkhard Monien. 1983. The Complexity of Determining Paths of Length k. In *Proceedings of the WG '83, International Workshop on Graphtheoretic Concepts in Computer Science*. 241–251.

[30] Marcin Mucha, Jesper Nederlof, Jakub Pawlewicz, and Karol Węgrzycki. 2019. Equal-Subset-Sum Faster Than the Meet-in-the-Middle. In *27th Annual European Symposium on Algorithms, ESA 2019*.

[31] Jesper Nederlof. 2016. Finding Large Set Covers Faster via the Representation Method. In *24th Annual European Symposium on Algorithms, ESA 2016*.

[32] Jesper Nederlof. 2020. Algorithms for NP-Hard Problems via Rank-related Parameters of Matrices. In *Festschrift Dedicated to the 60th Birthday of Hans Bodlaender*. Springer.

[33] Jesper Nederlof, Jakub Pawlewicz, Céline M.F. Swennenhuis, and Karol Węgrzycki. 2021. A Faster Exponential Time Algorithm for Bin Packing With a Constant Number of Bins via Additive Combinatorics. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 1682–1701.

[34] Jesper Nederlof, Erik Jan van Leeuwen, and Ruben van der Zwaan. 2012. Reducing a Target Interval to a Few Exact Queries. In *Mathematical Foundations of Computer Science 2012 - 37th International Symposium, MFCS 2012*. 718–727.

[35] Jesper Nederlof and Karol Węgrzycki. 2020. Improving Schroeppel and Shamir's Algorithm for Subset Sum via Orthogonal Vectors. (2020). arXiv:2010.08576 https://arxiv.org/abs/2010.08576

[36] Anup Rao and Amir Yehudayoff. 2020. *Communication Complexity: and Applications*. Cambridge University Press.

[37] Richard Schroeppel and Adi Shamir. 1981. A T=$O(2^{n/2})$, S=$O(2^{n/4})$ Algorithm for Certain NP-Complete Problems. *SIAM J. Comput.* 10, 3 (1981), 456–464.

[38] Virginia Vassilevska-Williams. 2018. On Some Fine-Grained Questions in Algorithms and Complexity. In *Proceedings of the International Congress of Mathematicians (ICM 2018)*. 3447–34.

[39] Ryan Williams. 2005. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.* 348, 2-3 (2005), 357–365.

[10] Additionally they derandomize their construction by using brute-force to find the probabilistic construction and use splitters to derandomize the step of splitting the universe into c blocks.