Subset Sum via Partial Match

Baitian Li* Lixi Ye†

Abstract

We show that the subset sum problem can be solved in randomized time $O(2^{0.49999\overline{7}n})$, breaking the $2^{n/2}$ meet-in-the-middle barrier.

Our approach builds on the representation technique framework of Randolph and Węgrzycki (STOC 2026). We choose a representation for which testing the compatibility of pairs of partial solution vectors is exactly the *partial match* problem. To beat exponent 1/2 for subset sum, it then suffices to give a nontrivial partial match algorithm in a certain parameter regime. We achieve this by designing a depth-2 linear circuit for the partial match matrix, which yields an efficient algorithm via the framework of Alman and Li (FOCS 2025).

Declaration. A result of a similar type was recently released by OpenAI: *Subset Sum in Time* $O(2^{0.49n})$ [Ope26], dated 4 October 2026. Our result was developed independently of that work and uses a different approach; see Section 1.4 for a brief comparison of the two approaches and AI disclosure and timestamps at the end of the paper for details.

*Columbia University. bl3052@columbia.edu. Supported in part by NSF Grant CCF-2238221 and a Packard Foundation Fellowship.

†Institute for Interdisciplinary Information Sciences, Tsinghua University. ylx25@mails.tsinghua.edu.cn.

1 Introduction

Given n integers and a target, the *subset sum* problem asks whether some subset of the integers sums to the target. Formally, given a vector $\mathbf{z} \in \mathbb{Z}^n$ and a target $\boldsymbol{\tau} \in \mathbb{Z}$, it asks for a vector $\mathbf{c} \in \{0,1\}^n$ such that $\mathbf{c} \cdot \mathbf{z} = \boldsymbol{\tau}$.

In their foundational work, Horowitz and Sahni [HS74] gave an $O^*(2^{n/2})$-time algorithm for subset sum,[1] a method now known as *meet-in-the-middle*. Chen, Jin, Randolph, and Servedio [CJRS23] later gave a randomized $O(2^{n/2}n^{-\gamma})$-time algorithm with $\gamma \approx 0.5023$, in the word RAM model. Whether subset sum can be solved *exponentially* faster, say in time $O(2^{(1/2-\varepsilon)n})$ for some constant $\varepsilon > 0$, has been a longstanding open question in exact algorithms.

Howgrave-Graham and Joux [HGJ10] were the first to break the meet-in-the-middle barrier in an *average-case* setting of subset sum, with running time $O(2^{0.337n})$. Their algorithm was subsequently improved in [BCJ11, BBSS20]. Although these algorithms are not worst-case algorithms, [HGJ10] introduced the *representation technique*, which later proved very fruitful. Mucha, Nederlof, Pawlewicz, and Węgrzycki [MNPW19] showed that the representation technique also yields a faster worst-case algorithm for a related problem, *equal subset sum*, in which the coefficient vector ranges over $\mathbf{c} \in \{-1,0,1\}^n \setminus \{\mathbf{0}\}$ and the target is 0. Compared with the meet-in-the-middle bound $O^*(3^{n/2}) \subseteq O(1.7321^n)$, they achieved time $O(1.7088^n)$.

Recently, Randolph and Węgrzycki [RW26] systematically studied a class of problems related to subset sum, which we call *generalized subset sum* (GSS). Here the target is 0, and the coefficient vector ranges over C^n or $C^n \setminus \{\mathbf{0}\}$ for a fixed finite set $C \subseteq \mathbb{Z}$ that defines the problem. Their starting point is the earlier work of Nederlof and Węgrzycki [NW21], which used an algorithm for detecting sparse Orthogonal Vectors to improve the space complexity of the $O^*(2^{n/2})$-time, $O^*(2^{n/4})$-space subset sum algorithm of Schroeppel and Shamir [SS81] to $O^*(2^{(1/4-\varepsilon)n})$, while keeping the time $O^*(2^{n/2})$. Randolph and Węgrzycki observed that this Orthogonal Vectors subroutine generalizes to a wider family of problems, which they call *compatibility testing*. Combining compatibility testing with the representation technique, they showed that GSS can be solved in time $O(|C|^{(1/2-\varepsilon)n})$ for a large family of sets C, beating meet-in-the-middle. In particular, they beat meet-in-the-middle for $C = \{-d, \dots, d\}$ for every $d > 1$, and for $C = [\pm d] \coloneqq \{-d, \dots, -1, 1, \dots, d\}$ for every $d > 2$. They also improved the equal subset sum bound of [MNPW19] to $O^*(1.7067^n)$. For $C = [\pm 1]$, GSS is equivalent to subset sum via the affine change of coefficients $\mathbf{c} \mapsto 2\mathbf{c} - \mathbf{1}$. This leaves the tantalizing possibility that their method extends to subset sum itself.

In this paper, we show that this line of approach indeed solves subset sum in time below $2^{n/2}$.

Theorem 1.1 (Theorem 4.1)**.** *Subset sum can be solved in randomized time* $O(2^{0.49999\overline{7}n})$.[2]

1.1 The representation technique

We first describe the framework of the representation technique [HGJ10, MNPW19, RW26], on which our algorithm builds.

The starting point of the representation technique is to carefully choose sets $\mathcal{A}, \mathcal{B} \subseteq \mathbb{Z}^n$ and then search for $\mathbf{a} \in \mathcal{A}$ and $\mathbf{b} \in \mathcal{B}$ such that $\mathbf{c} = \mathbf{a} + \mathbf{b}$ is a solution. For example, the classical meet-in-the-middle algorithm takes

$$\mathcal{A} = \{0,1\}^{n/2} \times 0^{n/2}, \qquad \mathcal{B} = 0^{n/2} \times \{0,1\}^{n/2}.$$

At first glance, this approach cannot break the $2^{n/2}$ barrier: to cover all 2^n candidate solutions, we need $|\mathcal{A}| \cdot |\mathcal{B}| \geq 2^n$, so at least one of $\mathcal{A}$ and $\mathcal{B}$ has size at least $2^{n/2}$. The key observation of the

[1] We use $O^*(\cdot)$ to suppress polynomial factors in n and the input length (measured in bits).

[2] Here $0.49999\overline{7} = 0.49999777\ldots = 1/2 - 1/450000$.

representation technique [HGJ10], as used in [MNPW19], is that one need not list $\mathbf{a} \cdot \mathbf{z}$ and $\mathbf{b} \cdot \mathbf{z}$ for all $\mathbf{a} \in \mathcal{A}$ and $\mathbf{b} \in \mathcal{B}$. The idea is that, instead of making $\mathcal{A}$ and $\mathcal{B}$ as small as possible, one should first enlarge them: the benefit is that each solution $\mathbf{c}$ then admits many representations $\mathbf{c} = \mathbf{a} + \mathbf{b}$ with $\mathbf{a} \in \mathcal{A}$ and $\mathbf{b} \in \mathcal{B}$.

Mixing and hashing. Suppose that the choice of $\mathcal{A}$ and $\mathcal{B}$ guarantees that every solution $\mathbf{c}$ has at least N_{rep} representations $(\mathbf{a}, \mathbf{b}) \in \mathcal{A} \times \mathcal{B}$. Following [AKKN16, MNPW19, RW26], one first analyzes the algorithm under an idealized *mixing* assumption, and then complements this analysis with a *mixing dichotomy*: whenever the mixing assumption fails, the instance can be solved by a different, faster algorithm. The algorithm of [RW26], as well as ours, relies on a more technically involved form of this dichotomy. Nonetheless, throughout this overview the reader may keep in mind the simplest form of the mixing assumption, which is the easiest to analyze: the sums $\mathbf{a} \cdot \mathbf{z}$ are *pairwise distinct* over all representations $\mathbf{a} + \mathbf{b} = \mathbf{c}$.

In this case, one can avoid listing all $|\mathcal{A}| + |\mathcal{B}|$ sums as follows. Pick a random prime p of size near N_{rep}, and then a uniformly random residue r modulo p.

1. List all $\mathbf{a} \in \mathcal{A}$ with $\mathbf{a} \cdot \mathbf{z} \equiv r \pmod{p}$ and all $\mathbf{b} \in \mathcal{B}$ with $\mathbf{b} \cdot \mathbf{z} \equiv \boldsymbol{\tau} - r \pmod{p}$; denote these sets by $\boldsymbol{A}$ and $\boldsymbol{B}$, respectively.

2. Find $\mathbf{a} \in \boldsymbol{A}$ and $\mathbf{b} \in \boldsymbol{B}$ such that $\mathbf{a} + \mathbf{b}$ is a solution.

With suitable hashing parameters and repetitions, a representation survives with high probability. In each trial, the expected filtered list sizes are roughly $|\mathcal{A}|/N_{\text{rep}}$ and $|\mathcal{B}|/N_{\text{rep}}$, up to polynomial factors.

For the choices of $\mathcal{A}$ and $\mathcal{B}$ below, $\boldsymbol{A}$ and $\boldsymbol{B}$ can be listed in time roughly $|\boldsymbol{A}| + |\boldsymbol{B}|$, together with the cost of generating suitable partial lists, so this part can run well below $2^{n/2}$. However, another technical problem arises: if $\mathcal{A} + \mathcal{B} \not\subseteq \{0,1\}^n$, then step 2 becomes harder, since one must sift out the *pseudosolutions*, i.e., pairs with $\mathbf{a} \cdot \mathbf{z} + \mathbf{b} \cdot \mathbf{z} = \boldsymbol{\tau}$ but $\mathbf{a} + \mathbf{b} \notin \{0,1\}^n$.

The insight of [RW26] is that a carefully optimized algorithm for detecting true solutions can be used to speed up certain generalized subset sum problems. As an illustrative application, consider the case where the coefficient vector $\mathbf{c}$ may range over $\{-1,0,1\}^n$. In [MNPW19], the authors showed how to beat the meet-in-the-middle bound $3^{n/2}$ by taking $\mathcal{A} = \{0,1\}^n$ and $\mathcal{B} = \{0,-1\}^n$. Then [RW26] improved this exponent by taking $\mathcal{A}$ and $\mathcal{B}$ to be carefully chosen subsets of larger sets,

$$\mathcal{A} \subseteq \{0,1,2\}^n, \qquad \mathcal{B} \subseteq \{-2,-1,0\}^n.$$

The values of $a_i + b_i$ are given by the following table:

$a_i \backslash b_i$	-2	-1	0
0	-2	$\mathbf{-1}$	$\mathbf{0}$
1	$\mathbf{-1}$	$\mathbf{0}$	$\mathbf{1}$
2	$\mathbf{0}$	$\mathbf{1}$	2

where the entries in bold are those with $a_i + b_i \in \{-1,0,1\}$. Since a true solution must lie in $\{-1,0,1\}^n$, the relevant compatibility matrix is

$$P_{\text{RW}} = \begin{array}{c|ccc} a_i \backslash b_i & -2 & -1 & 0 \\ \hline 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 2 & 1 & 1 & 0 \end{array}.$$

Sifting out pseudosolutions in step 2 thus leads to the following algorithmic problem: given sets of vectors $\boldsymbol{A} \subseteq \{0,1,2\}^n$ and $\boldsymbol{B} \subseteq \{-2,-1,0\}^n$, find $\mathbf{a} \in \boldsymbol{A}$ and $\mathbf{b} \in \boldsymbol{B}$ such that the corresponding entry of $P_{\mathrm{RW}}^{\otimes n}$ is 1, i.e., $(P_{\mathrm{RW}})_{a_i,b_i} = 1$ for every i. Randolph and Węgrzycki [RW26] call this problem *compatibility testing*; they designed a fast compatibility testing algorithm for P_{RW}, which improves the exponent obtained in [MNPW19]. One of our key technical contributions is a different choice of representation, whose compatibility matrix admits a surprisingly fast testing algorithm.

1.2 Algorithm overview

We now describe our algorithm at a level of detail sufficient to verify its running time; Sections 2–4 give the formal proof. Throughout, $\mathrm{h}(t) := -t\log_2 t - (1-t)\log_2(1-t)$ denotes the binary entropy function, with $0\log_2 0 := 0$.

The representation and its compatibility matrix. We represent $\mathbf{c} \in \{0,1\}^m$ as $\mathbf{c} = \mathbf{a} + \mathbf{b}$ with $\mathbf{a} \in \{0,1\}^m$ and $\mathbf{b} \in \{-1,0,1\}^m$, using

$$0 = 0+0 = 1+(-1), \qquad 1 = 0+1 = 1+0.$$

For $s \in [0,1]$, let $T_{m,s} \subseteq \{-1,0,1\}^m$ be the set of vectors with exactly sm nonzero entries. The values of $a_i + b_i$ are given by the following table:

$a_i \backslash b_i$	-1	0	1
0	-1	$\mathbf{0}$	$\mathbf{1}$
1	$\mathbf{0}$	$\mathbf{1}$	2

where the entries in bold are those with $a_i + b_i \in \{0,1\}$. Accordingly, the compatibility matrix is

$$P = \begin{array}{c|ccc} a_i \backslash b_i & -1 & 0 & 1 \\ \hline 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{array},$$

and compatibility testing asks, given $\boldsymbol{A} \subseteq \{0,1\}^m$ and $\boldsymbol{B} \subseteq T_{m,s}$, whether some $\mathbf{a} \in \boldsymbol{A}$ and $\mathbf{b} \in \boldsymbol{B}$ satisfy $\mathbf{a} + \mathbf{b} \in \{0,1\}^m$, i.e., whether the corresponding entry of $P^{\otimes m}$ is 1.

Relabeling the columns via $1 \mapsto \mathtt{0}$, $-1 \mapsto \mathtt{1}$, and $0 \mapsto *$, and ordering them as $\mathtt{0}, \mathtt{1}, *$, turns P into the partial match matrix

$$P_1 = \begin{array}{c|ccc} & \mathtt{0} & \mathtt{1} & * \\ \hline \mathtt{0} & 1 & 0 & 1 \\ \mathtt{1} & 0 & 1 & 1 \end{array},$$

where $*$ is a wildcard matching both $\mathtt{0}$ and $\mathtt{1}$. We call strings over $\{0,1\}$ *binary strings* and strings over $\{0,1,*\}$ *query strings* (or queries). Under this relabeling, $T_{m,s}$ becomes the set of queries with exactly $(1-s)m$ wildcards. Thus compatibility testing for our representation is exactly partial match. A key technical ingredient of our algorithm is the following lemma for the "half-stars" case $s = 1/2$.

Lemma 1.2. *There are nonnegative constants l, r and a constant $\xi > 0$ with $l + r \le 1/2 - 2\xi$ such that, for every $\varepsilon > 0$, there is a deterministic algorithm which, for even m, tests compatibility for $s = 1/2$ in time*

$$O\Big(2^{(l+\varepsilon)m}|\boldsymbol{A}| + 2^{(r+\varepsilon)m}|\boldsymbol{B}|\Big).$$

Other support densities $s < 1/2$ reduce to the case $s = 1/2$ by a simple padding argument: we append $(1-2s)m$ dummy pairs $(1,-1)$, which are always compatible, to every left and right vector. After relabeling, the padded strings have even length $(2-2s)m$, and exactly half of the positions of each query are specified. Lemma 1.2 then tests compatibility in time

$$O\Big(2^{2(1-s)(l+\varepsilon)m}|\boldsymbol{A}| + 2^{2(1-s)(r+\varepsilon)m}|\boldsymbol{B}|\Big)\,. \tag{1.1}$$

If "$\xi > 0$" is relaxed to "$\xi \geq 0$", the lemma would be trivial: the pair $(l, r) = (0, 1/2)$ is achieved by letting each query enumerate all $2^{m/2}$ completions of its wildcards. As we show below, beating this trivial bound $l + r = 1/2$ by any positive constant already yields a faster subset sum algorithm.

The search space. We use representations only on the first $n/3$ coordinates, which we call the *prefix*, and plain meet-in-the-middle on the remaining $2n/3$; this lets us balance the asymmetric costs of the two sides. For a parameter ℓ chosen below and a support density $s \in [0, 1/2]$, let

$$\mathcal{A} = \{0,1\}^{n/3} \times \{0,1\}^{\ell} \times 0^{2n/3-\ell}, \qquad \mathcal{B} = T_{n/3,s} \times 0^{\ell} \times \{0,1\}^{2n/3-\ell},$$

so that

$$|\mathcal{A}| = 2^{n/3+\ell}, \qquad |\mathcal{B}| = \binom{n/3}{sn/3} 2^{sn/3+2n/3-\ell}.$$

We run the algorithm for every possible value of s.

Mixing case: many correct keys. Fix a solution $\mathbf{c}$. The representations $\mathbf{c} = \mathbf{a} + \mathbf{b}$ with $(\mathbf{a}, \mathbf{b}) \in \mathcal{A} \times \mathcal{B}$ differ only on the prefix, and each binary prefix of $\mathbf{a}$ determines $\mathbf{b} = \mathbf{c} - \mathbf{a}$. Let N_{sum} be the number of distinct subset sums of the first $n/3$ input numbers. The mixing assumption discussed above corresponds to the perfect case $N_{\text{sum}} = 2^{n/3}$, in which all representations of $\mathbf{c}$ have distinct keys $\mathbf{a} \cdot \mathbf{z}$. We replace it by the weaker assumption that N_{sum} is large, and determine below how large it needs to be.

We first observe that some support layer contains many distinct keys. Pair each binary prefix of $\mathbf{a}$ with its complement $\mathbf{a}'$. The corresponding vectors $\mathbf{b} = \mathbf{c} - \mathbf{a}$ and $\mathbf{b}' = \mathbf{c} - \mathbf{a}'$ have prefix support sizes k and $n/3 - k$, respectively, and the prefix sums of $\mathbf{a}$ and $\mathbf{a}'$ add up to the total prefix sum, which is fixed. Hence at least $N_{\text{sum}}/2$ distinct prefix sums have a representation with prefix support size at most $n/6$, so some layer $s \leq 1/2$ contains $\Omega(N_{\text{sum}}/n)$ distinct correct keys.

Since we run over all s, it suffices for the algorithm to succeed in this layer. We hash the keys modulo a random prime of size near N_{sum}, as described above.[3] The surviving lists $\boldsymbol{A}, \boldsymbol{B}$ then have expected sizes roughly

$$\mathbb{E}[|\boldsymbol{A}|] = O\left(\frac{|\mathcal{A}|}{N_{\text{sum}}}\right), \qquad \mathbb{E}[|\boldsymbol{B}|] = O\left(\frac{|\mathcal{B}|}{N_{\text{sum}}}\right),$$

and they can be generated from meet-in-the-middle partial lists in time close to their sizes.

We then run the padded compatibility testing algorithm (1.1) with $m = n/3$ on the prefixes of the surviving vectors, grouping them by their complete integer key so that compatibility and the target-sum condition are checked together. Using $\binom{n/3}{sn/3} \leq 2^{\mathrm{h}(s)n/3}$, for any fixed $\varepsilon > 0$, the

[3] We remark that the formal proof uses a slightly different hash function, which avoids some technicalities arising when the input integers have up to $2^{o(n)}$ bits.

expected cost is, up to subexponential factors,

$$\frac{2^{2(1-s)(l+\varepsilon)n/3}|\mathcal{A}| + 2^{2(1-s)(r+\varepsilon)n/3}|\mathcal{B}|}{N_{\text{sum}}} \leq \frac{2^{2(1-s)(l+\varepsilon)n/3+n/3+\ell} + 2^{2(1-s)(r+\varepsilon)n/3+(\mathrm{h}(s)+s)n/3+2n/3-\ell}}{N_{\text{sum}}}.$$

Choosing

$$\ell \coloneqq \frac{n}{6}\left(1 + \mathrm{h}(s) + s - 2(l-r)(1-s)\right)$$

balances the two terms, and their common exponent is

$$\frac{n}{2} + \left((l+r+2\varepsilon)(1-s) + \frac{\mathrm{h}(s)+s}{2}\right)\frac{n}{3} - \log_2 N_{\text{sum}}.$$

Using $\mathrm{h}(s) \leq 1$, $l + r \leq 1/2 - 2\xi$, and $s \leq 1/2$ gives the upper bound

$$\frac{n}{2} + \frac{(1-\xi)n}{3} - \log_2 N_{\text{sum}} + \frac{2\varepsilon n}{3}.$$

Consequently, whenever $N_{\text{sum}} \geq 2^{(1-\xi+\eta)n/3}$ for some constant $\eta > 0$, choosing $\varepsilon < \eta/2$ beats $2^{n/2}$ by an exponential factor.

Mixing dichotomy. When N_{sum} is small, we instead use the few-sums meet-in-the-middle procedure of [AKKN16], which runs in time $\sqrt{N_{\text{sum}}}2^{n/3}$. For a threshold γ,

- if $N_{\text{sum}} \leq 2^{(1-\gamma)n/3}$, the procedure of [AKKN16] takes time at most $2^{(1/2-\gamma/6)n+o(n)}$;
- otherwise, the many-sums procedure above takes time at most $2^{(1/2-(\xi-\gamma)/3+2\varepsilon/3)n+o(n)}$.

Choosing $\gamma \coloneqq 2\xi/3$ makes both exponents before the added slack $1/2 - \xi/9$. Accounting for the partial match algorithm's exponent slack, subset sum can be solved in time $O(2^{(1/2-\xi/9+\varepsilon)n})$ for every $\varepsilon > 0$. Our circuit gives $\xi > 1/50000$, leaving positive slack before the exponent $1/2 - 1/450000$ of Theorem 4.1.

1.3 Partial match via depth-2 linear circuits

Finally, we briefly explain how we obtain the partial match algorithm of Lemma 1.2.

Depth-2 linear circuits. Let $M \in \{0,1\}^{I \times J}$ be a large, implicitly given 01 matrix. Given $\boldsymbol{A} \subseteq I$ and $\boldsymbol{B} \subseteq J$, with sizes much smaller than $|I|$ and $|J|$, respectively, we want to decide whether some $(a,b) \in \boldsymbol{A} \times \boldsymbol{B}$ has $M_{a,b} = 1$; this is compatibility testing for M.

A depth-2 linear circuit for M is simply a matrix factorization $M = \mathbf{LR}$. It can be viewed as a circuit computing $\mathbf{u} \mapsto \mathbf{uLR}$: the input gates carry the coordinates of $\mathbf{u}$, each middle gate computes a linear combination of the values at the input gates, and each output gate computes a linear combination of the values at the middle gates. The relevant measure here is the pair of *degrees*: the circuit has degrees $(\deg_{\mathsf{L}}, \deg_{\mathsf{R}})$ if every row of $\mathbf{L}$ has at most $\deg_{\mathsf{L}}$ nonzero entries and every column of $\mathbf{R}$ has at most $\deg_{\mathsf{R}}$ nonzero entries.

Given such a circuit, a simple algorithm solves compatibility testing in time $O(|\boldsymbol{A}| \deg_{\mathsf{L}} + |\boldsymbol{B}| \deg_{\mathsf{R}})$, up to the cost of arithmetic. Writing $M_{a,b} = \mathbf{L}_{a,\cdot} \cdot \mathbf{R}_{\cdot,b}$, we obtain

$$\sum_{a \in \boldsymbol{A},\, b \in \boldsymbol{B}} M_{a,b} = \Big(\sum_{a \in \boldsymbol{A}} \mathbf{L}_{a,\cdot}\Big) \cdot \Big(\sum_{b \in \boldsymbol{B}} \mathbf{R}_{\cdot,b}\Big),$$

and by the degree bounds, the two vectors on the right have at most $|\boldsymbol{A}| \deg_{\mathsf{L}}$ and $|\boldsymbol{B}| \deg_{\mathsf{R}}$ nonzero entries, respectively. When the factorization holds over the integers, the left-hand side is exactly the number of compatible pairs; we call such circuits $\mathsf{EXACT}_{\mathbb{Z}}$ circuits.

Many known compatibility testing algorithms implicitly construct such a circuit, with the caveat that their arithmetic is over the Boolean semiring rather than the integers; we call these OR circuits. For example, the Orthogonal Vectors algorithm of [NW21] covers the Orthogonal Vectors matrix (also known as the disjointness matrix) by combinatorial rectangles, each of which is a rank-one 01 matrix represented by a single middle gate.

Uniformization. Proving the mere *existence* of a good depth-2 linear circuit would a priori yield only a *non-uniform* algorithm, since a uniform algorithm must also *efficiently construct* the circuit. For depth-2 linear circuits, this is not a genuine obstacle, thanks to the *tensor product*: it lets us start from a depth-2 linear circuit of *fixed* size and bootstrap it into an explicit family that handles arbitrarily large n.

This phenomenon is sometimes called *uniformization*. Williams [Wil24, Theorem 8] showed that a notion called *equality rank* admits uniformization, and combined this with a fixed-size construction for the disjointness matrix to obtain Orthogonal Vectors algorithms. Perhaps the most famous instance of uniformization is matrix multiplication: Strassen [Str69] found a bilinear algorithm of rank 7 for 2×2 matrix multiplication, and applying it recursively turns this single finite construction into a uniform algorithm for all dimensions.

Alman and Li [AL25] observed that an equality rank bound implies a degree bound for depth-2 linear circuits. Their framework, which turns a depth-2 linear circuit for a matrix into an algorithm, thus generalizes Williams's approach, and it is the framework we build on.

Rebalancing. Uniformization therefore lets us concentrate on bounding the degrees of circuits; explicit constructions, and hence algorithms, follow automatically. Bounding the degrees directly is nevertheless difficult, so we take a two-step route:

(1) construct a family of circuits with complementary advantages, none of which need meet our degree requirement on its own; and

(2) combine their advantages to obtain a genuine improvement in the degrees.

For step (2), previous works on depth-2 linear circuits [JS13, AGP23, Ser22, AL25] developed a technique called *rebalancing*. These works use rebalancing to control the circuit *size*, whereas we need to control the *degrees*. Fortunately, [AL25] gives a general analysis of a certain rebalancing process, which adapts to our setting after a slight technical modification.

The partial match matrix. For our algorithmic purposes, we need to uniformly construct circuits for the matrix $M = P^{\otimes n}$ with its columns restricted to $T_{n,s}$. After the relabeling above, this is the matrix $P_{n,s}$, which we call a *partial match matrix*.

The rest of the construction is technically involved. In the remainder of this overview, we only sketch the main ideas and their relation to previous work, deferring the details to Section 3.

A minor technical issue is that $P_{n,s}$ is not itself a tensor power, only a submatrix of the tensor power $P^{\otimes n}$, so the general uniformization results for tensor powers do not apply directly. We overcome this by noting that $P_{n,1/2}$ is still *nearly* a tensor power: up to coordinate permutations, it can be covered by a small number of copies of a tensor power of a fixed-size partial match matrix $P_{k,1/2}$, which is enough for our purpose.

Circuit construction. To explain the ingredients of our circuit construction, we first review a classical result of Fredman and Volper [FV82] on data structures for partial match, which can be implicitly viewed as an efficient PAR circuit for $P_{n,s}$.

There are two trivial circuits, given by the factorizations $P_{n,s} = P_{n,s} \cdot I$ and $P_{n,s} = I \cdot P_{n,s}$. In the first one, every string enumerates the $\binom{n}{sn}$ possible wildcard masks; in the second one, every query enumerates all $2^{(1-s)n}$ ways to fill its wildcards. The first one gives $l + r \le \mathrm{h}(s)$, and the second one gives $l + r \le 1 - s$.

In the regime $s \le 1/2$, Fredman and Volper combine the two circuits as follows.

(1) Pick a family $\mathcal{B}$ of subsets of $[n]$ of size $2sn$ such that every subset S of size sn is contained in some $B \in \mathcal{B}$. Assign to each S one such B.

(2) For each $B \in \mathcal{B}$, construct a circuit for the submatrix formed by the columns of the queries $\mathbf{y}$ whose set $S(\mathbf{y})$ of non-wildcard positions is assigned to B. This circuit has 2^{2sn} middle gates, corresponding to all assignments of values to the positions in B. Each string is linked to the unique assignment it agrees with, and each query is linked to the 2^{sn} assignments that agree with it on $S(\mathbf{y})$, that is, to all ways of filling the remaining positions $B \setminus S(\mathbf{y})$.

Since every query is handled by exactly one B, the right degree is 2^{sn}. Each B contributes 1 to the degree of every string, so the left degree is $|\mathcal{B}|$. By a classical probabilistic argument, all sets S can be covered by roughly $\binom{n}{sn}/\binom{2sn}{sn} \approx 2^{(\mathrm{h}(s)-2s)n}$ sets B.

This gives $l + r \le \mathrm{h}(s) - s$, which improves on both trivial circuits for every $0 < s < 1/2$. One can view this construction as using a *random covering* argument to combine the advantages of the two trivial factorizations. At $s = 1/2$, however, the only choice is $B = [n]$, and the bound degenerates to the trivial $l + r = 1/2$.

Interestingly, this construction, like the two trivial ones, is a PAR circuit, and in particular also an OR circuit. Although OR circuits are often the easier ones to construct, for example for the disjointness matrix [NW21, AL25], they do not help for the partial match matrix in our regime: a simple combinatorial argument shows that they cannot beat $l + r = 1/2$ (Theorem 3.10).

Therefore, to improve the case $s = 1/2$ that we are interested in, we need to genuinely exploit algebraic cancellation. Besides the two trivial factorizations, we introduce a third one, coming from the *inclusion–exclusion principle*. For a query $\mathbf{y}$, let $A_0(\mathbf{y})$ and $A_1(\mathbf{y})$ be the sets of positions where $\mathbf{y}$ is `0` and `1`, respectively. Then whether a binary string $\mathbf{x}$ matches $\mathbf{y}$ is given by the polynomial

$$\prod_{i \in A_1(\mathbf{y})} x_i \prod_{i \in A_0(\mathbf{y})} (1 - x_i) = \sum_{A_1(\mathbf{y}) \subseteq U \subseteq A_1(\mathbf{y}) \cup A_0(\mathbf{y})} (-1)^{|U| - |A_1(\mathbf{y})|} \mathbf{x}^U,$$

where $\mathbf{x}^U \coloneqq \prod_{i \in U} x_i$. Viewing each monomial $\mathbf{x}^U$ as a middle gate, the right-hand side is a new factorization of the partial match matrix, now with coefficients ± 1: the string $\mathbf{x}$ is linked to all subsets of its support, and the query $\mathbf{y}$ to all sets U in the above sum. Combining this factorization with a random covering argument and feeding the result into the rebalancing machinery, we can, perhaps surprisingly, beat $l + r = 1/2$.

1.4 Comparison with the OpenAI preprint

In this subsection, we briefly discuss the difference between our approach and the approach of the recent OpenAI preprint [Ope26]. The OpenAI preprint also follows the representation technique framework developed in [HGJ10, MNPW19, RW26]. The technical twist, however, is that their approach *bypasses* the compatibility testing problem altogether.

The high-level approach of the OpenAI preprint can be described as follows. First, pick a large random prime $p \approx 2^{(1/2+\varepsilon)n}$. Then use a procedure (1) to estimate a checksum over all modular solutions, that is, over all $\mathbf{c} \in \{0,1\}^n$ with $\mathbf{c} \cdot \mathbf{z} \equiv \boldsymbol{\tau} \pmod{p}$. Next, use another procedure (2) to find all $\mathbf{c}$ with $\mathbf{c} \cdot \mathbf{z} = \boldsymbol{\tau} + kp$ for some $k \in \mathbb{Z} \setminus \{0\}$; these are called *aliases* in the preprint. Finally, subtracting the contribution of the aliases from the checksum leaves the checksum of the integral solutions, and this detects whether an integral solution exists. We also note that, to make the algorithm work, procedure (1) actually only needs to estimate the checksum rather than compute it exactly; this step also contains technically involved ingredients.

Only procedure (2) uses the representation technique, and in their algorithm there is no need to design a faster compatibility testing algorithm. Recall that in the usual representation technique, compatibility testing is required to sift out the pseudosolutions when one wants to find a true solution, because there is no good bound on the number of pseudosolutions. In their setting, however, the randomness of p allows them to bound the number of pseudosolutions representing the aliases, so they can afford to enumerate all of these pseudosolutions.

Compared with the OpenAI result, which avoids compatibility testing, our improvement comes from tackling the compatibility testing problem directly. To achieve this, we use the machinery of depth-2 linear circuits, together with new constructions of such circuits.

1.5 Organization

Section 2 reduces subset sum to an algorithm for "half-stars" partial match.

Section 3 constructs such an algorithm from depth-2 linear circuits. Section 3.1 introduces our notation for depth-2 linear circuits. Section 3.2 establishes the precise form of uniformization that we need for the partial match matrices. Section 3.3 proves our degree version of rebalancing. Section 3.4 gives the circuit construction and bounds its degrees with the help of rebalancing. Finally, Section 3.5 shows that OR circuits cannot improve on the trivial bound $l + r = 1/2$; this justifies our use of algebraic cancellation.

Section 4 combines the two with explicit parameters to prove Theorem 4.1: subset sum can be solved by a randomized algorithm in worst-case time $O(2^{(1/2-1/450000)n})$.

2 Reduction

Formally, an instance of subset sum consists of integers $\mathbf{z} = (z_1, \ldots, z_n) \in \mathbb{Z}^n$ and a target $\boldsymbol{\tau} \in \mathbb{Z}$, and the task is to find $\mathbf{c} \in \{0,1\}^n$ with $\mathbf{c} \cdot \mathbf{z} = \boldsymbol{\tau}$, or to report that no such vector exists. We assume that the input length is $2^{o(n)}$ bits, and the running times of our algorithms are measured in bit operations.

In this section, we show that a sufficiently fast algorithm for a special case of partial match yields a subset sum algorithm beating $2^{n/2}$. Section 2.1 defines the partial match problem we need, and Section 2.2 gives the reduction. Throughout this section, $\mathrm{h}(t) \coloneqq -t \log_2 t - (1-t)\log_2(1-t)$ denotes the binary entropy function, with $0 \log_2 0 \coloneqq 0$.

2.1 Partial match

The compatibility testing problem arising from our representation is governed by the matrix

$$P_1 \coloneqq \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix},$$

which we call the *partial match* matrix. Its rows are indexed by $0, 1$ and its columns by $0, 1, *$.

We write $\otimes$ for the *Kronecker product* of matrices. In general, for matrices A with rows indexed by I and columns by J, and B with rows indexed by I' and columns by J', the Kronecker product $A \otimes B$ has rows indexed by $I \times I'$ and columns by $J \times J'$, and is defined by

$$(A \otimes B)_{(i,i'),(j,j')} \coloneqq A_{i,j} \cdot B_{i',j'}.$$

Thus the rows of $P_1^{\otimes n}$ are indexed by *binary strings* $\mathbf{x} \in \{0,1\}^n$, its columns by *query strings* $\mathbf{y} \in \{0,1,*\}^n$, and $(P_1^{\otimes n})_{\mathbf{x},\mathbf{y}} = \prod_{i=1}^n (P_1)_{x_i,y_i}$.

This notion lets us describe the matching of strings. For $\mathbf{y} \in \{0,1,*\}^n$, put $S(\mathbf{y}) \coloneqq \{i : y_i \neq *\}$, the set of *specified* positions of $\mathbf{y}$. Then $(P_1^{\otimes n})_{\mathbf{x},\mathbf{y}} = 1$ if and only if $x_i = y_i$ for every $i \in S(\mathbf{y})$, that is, $\mathbf{x}$ matches every specified position of $\mathbf{y}$; in this case we say that $\mathbf{x}$ *matches* $\mathbf{y}$.

The reduction only needs the special case of partial match in which every query has exactly half of its positions specified.

Definition 2.1 (Half-stars partial match)**.** In the *half-stars partial match* problem, we are given an even integer n and two sets

$$\boldsymbol{A} \subseteq \{0,1\}^n, \qquad \boldsymbol{B} \subseteq \{\mathbf{y} \in \{0,1,*\}^n : |S(\mathbf{y})| = n/2\}.$$

The task is to decide whether some $\mathbf{x} \in \boldsymbol{A}$ matches some $\mathbf{y} \in \boldsymbol{B}$, that is, whether $(P_1^{\otimes n})_{\mathbf{x},\mathbf{y}} = 1$ for some $(\mathbf{x},\mathbf{y}) \in \boldsymbol{A} \times \boldsymbol{B}$.

The name "half-stars" refers to the requirement that exactly half of the positions of every query are wildcards.

We measure an algorithm for this problem by the work it spends on each string of either set. We call a pair $(l,r) \in \mathbb{R}^2_{\geq 0}$ an *admissible exponent pair* if for every $\varepsilon > 0$ there is a deterministic algorithm solving half-stars partial match in time

$$O\left(2^{(l+\varepsilon)n}|\boldsymbol{A}| + 2^{(r+\varepsilon)n}|\boldsymbol{B}|\right).$$

Clearly admissibility is closed upward: if (l,r) is an admissible exponent pair, $l' \geq l$ and $r' \geq r$, then (l',r') is also admissible.

For example, $(0,1/2)$ is an admissible exponent pair: sort $\boldsymbol{A}$, and for each $\mathbf{y} \in \boldsymbol{B}$, enumerate the $2^{n/2}$ binary strings matching $\mathbf{y}$ and look each of them up in $\boldsymbol{A}$. The main result of this section, Theorem 2.2, shows that improving the sum $l+r$ below $1/2$ by any positive constant already beats $2^{n/2}$ for subset sum.

2.2 The reduction

Theorem 2.2. *Suppose (l,r) is an admissible exponent pair and $l + r \leq 1/2 - 2\xi$ for some $\xi > 0$. For every $\varepsilon > 0$, subset sum has a randomized algorithm in time $O(2^{(1/2-\xi/9+\varepsilon)n})$.*

We fix some notation. From now on, we assume that $6 \mid n$: appending at most five zeros to $\mathbf{z}$ achieves this, and changes neither the set of solutions on the original coordinates nor the running time by more than a constant factor. For a subset sum instance $(\mathbf{z},\boldsymbol{\tau})$, the *prefix* consists of the first $n/3$ coordinates, and N_{sum} denotes the number of distinct subset sums of $z_1,\ldots,z_{n/3}$. For $\mathbf{a},\mathbf{b} \in \mathbb{Z}^n$, the *left key* of $\mathbf{a}$ is $\mathbf{a} \cdot \mathbf{z}$ and the *right key* of $\mathbf{b}$ is $\boldsymbol{\tau} - \mathbf{b} \cdot \mathbf{z}$. Thus $\mathbf{a}+\mathbf{b}$ is a solution if and only if the two keys are equal and $\mathbf{a}+\mathbf{b} \in \{0,1\}^n$.

We will technically use the following hash family [Die96]. For a power of two p, consider the family

$$\mathcal{H}_p \coloneqq \{h_{\alpha,\beta} : \alpha,\beta \in \mathbb{Z}_p\}, \qquad h_{\alpha,\beta}(\mathsf{k}) \coloneqq (\alpha\mathsf{k} + \beta) \bmod p \qquad (\mathsf{k} \in \mathbb{Z}).$$

For a power of two $q \le p$, put $\mathcal{I}_q := \{0, \dots, q-1\} \subseteq \mathbb{Z}_p$. We filter an integer k by keeping it if and only if $h(\mathsf{k}) \in \mathcal{I}_q$, where h is drawn uniformly from $\mathcal{H}_p$, that is, α and β are independent and uniform in $\mathbb{Z}_p$.

Lemma 2.3. *Let $q \le p$ be powers of two, and let h be uniform in $\mathcal{H}_p$.*

1. *For every $\mathsf{k} \in \mathbb{Z}$, we have $\Pr[h(\mathsf{k}) \in \mathcal{I}_q] = q/p$.*
2. *For all $\mathsf{k}_1, \mathsf{k}_2 \in \mathbb{Z}$ with $0 < |\mathsf{k}_1 - \mathsf{k}_2| < q$, the events $h(\mathsf{k}_1) \in \mathcal{I}_q$ and $h(\mathsf{k}_2) \in \mathcal{I}_q$ are independent.*

Proof of Lemma 2.3. For the first claim, $h_{\alpha,\beta}(\mathsf{k})$ is uniform in $\mathbb{Z}_p$ for every fixed α, since β is uniform.

For the second claim, put $g := \gcd(\mathsf{k}_2 - \mathsf{k}_1, p)$. Then $g \le |\mathsf{k}_2 - \mathsf{k}_1| < q$, and $g \mid q$ since both are powers of two. Conditional on $h_{\alpha,\beta}(\mathsf{k}_1) = t$, the multiplier α remains uniform, so $h_{\alpha,\beta}(\mathsf{k}_2) = t + \alpha(\mathsf{k}_2 - \mathsf{k}_1)$ is uniform on the coset $t + g\mathbb{Z}_p$. Exactly q/g of its p/g elements lie in $\mathcal{I}_q$. Hence $\Pr[h(\mathsf{k}_2) \in \mathcal{I}_q \mid h(\mathsf{k}_1) = t] = q/p$ for every t. □

Proof of Theorem 2.2. We first enumerate all $2^{n/3}$ subset sums of the prefix by brute force, in time $2^{n/3+o(n)}$. This determines N_{sum}, and we retain one representative prefix for each of the N_{sum} distinct sums. Depending on the value of N_{sum}, we then run one of the following two procedures.

- *Few distinct sums* [AKKN16, Lemma 3.2], when $N_{\text{sum}} \le 2^{(1-2\xi/3)n/3}$. Combine the representatives with $\ell := \lfloor n/3 - \log_2 N_{\text{sum}}/2 \rfloor$ outside coordinates, and apply meet-in-the-middle [HS74] against the remaining coordinates. The two lists have sizes $N_{\text{sum}} 2^{\ell}$ and $2^{2n/3-\ell}$. Including preprocessing and integer arithmetic, the time is
$$2^{o(n)} 2^{n/2} \sqrt{\frac{N_{\text{sum}}}{2^{n/3}}}.$$
- *Many distinct sums*, when $N_{\text{sum}} > 2^{(1-2\xi/3)n/3}$. This is the main case, which occupies the rest of the proof.

Representation. As in the representation technique [HGJ10, AKKN16, MNPW19, RW26], we search for $\mathbf{c} = \mathbf{a} + \mathbf{b}$, but we use our representations only on the prefix. For $s \in \{0, 3/n, \dots, 1/2\}$ and a parameter ℓ determined below, $\mathbf{a}$ and $\mathbf{b}$ range over the conceptual lists

$$\mathcal{A} := \{0,1\}^{n/3} \times \{0,1\}^{\ell} \times 0^{2n/3-\ell}, \qquad \mathcal{B} := T_{n/3,s} \times 0^{\ell} \times \{0,1\}^{2n/3-\ell},$$

where $T_{m,s} \subseteq \{-1,0,1\}^m$ is the set of vectors with exactly sm nonzero entries. We run the procedure for every such s, and we call the two blocks of coordinates after the prefix, of lengths ℓ and $2n/3 - \ell$, the *outside blocks*. Only the elements of $\mathcal{A}$ and $\mathcal{B}$ passing the filter below, which we call *records*, will be actually generated.

Next, we translate compatibility into partial match. In a prefix coordinate, $a_i + b_i \in \{0,1\}$ precisely when a_i matches the query symbol $1, *, 0$ for $b_i = -1, 0, 1$, respectively. On the outside coordinates, $\mathbf{a}$ and $\mathbf{b}$ have disjoint supports, so $\mathbf{a} + \mathbf{b}$ is always binary there.

To make the queries half-stars, prepend $(1-2s)n/3$ dummy pairs $(1,-1)$, which always match. For $\mathbf{a} \in \mathcal{A}$ and $\mathbf{b} \in \mathcal{B}$, write $\mathbf{x}(\mathbf{a})$ and $\mathbf{y}(\mathbf{b})$ for the resulting binary and query strings. They have even length $n' := (2-2s)n/3$, exactly $(1-s)n/3 = n'/2$ positions of $\mathbf{y}(\mathbf{b})$ are specified, and

$$\mathbf{a} + \mathbf{b} \in \{0,1\}^n \iff \mathbf{x}(\mathbf{a}) \text{ matches } \mathbf{y}(\mathbf{b}).$$

For strings of length n', the partial match algorithm runs in time $O(\mathrm{cost}(|\boldsymbol{A}|, |\boldsymbol{B}|))$, where

$$\mathrm{cost}(N_{\mathsf{L}}, N_{\mathsf{R}}) \coloneqq 2^{(l+\varepsilon)n'} N_{\mathsf{L}} + 2^{(r+\varepsilon)n'} N_{\mathsf{R}}.$$

We choose $\ell \in [0, n/2]$ as the largest even integer that is not greater than

$$\frac{n}{6}\left(1 + \mathrm{h}(s) + s - 2(l-r)(1-s)\right).$$

This is possible since $|l - r| \le 1/2$, so the expression itself lies in $[0, n/2]$. Since $6 \mid n$, both of the outside blocks must have even length. The following lemma bounds the cost of running partial match directly on all of $\mathcal{A}$ and $\mathcal{B}$, which we call the *unscreened* cost.

Lemma 2.4. *We have*

$$\mathrm{cost}(|\mathcal{A}|, |\mathcal{B}|) \le 2^{n/2 + (1-\xi)n/3 + 2\varepsilon n/3 + O(1)}.$$

Proof of Lemma 2.4. Using $\binom{n/3}{sn/3} \le 2^{\mathrm{h}(s)n/3}$, the list sizes are

$$|\mathcal{A}| = 2^{n/3+\ell}, \qquad |\mathcal{B}| = \binom{n/3}{sn/3} 2^{sn/3 + 2n/3 - \ell} \le 2^{(\mathrm{h}(s)+s)n/3 + 2n/3 - \ell}.$$

With $n' = 2(1-s)n/3$, the two terms of the cost thus have exponents at most

$$2(1-s)(l+\varepsilon)\frac{n}{3} + \frac{n}{3} + \ell \qquad \text{and} \qquad 2(1-s)(r+\varepsilon)\frac{n}{3} + (\mathrm{h}(s)+s)\frac{n}{3} + \frac{2n}{3} - \ell.$$

Our choice of ℓ makes them equal up to an additive constant, so both are at most their average plus $O(1)$. Using $\mathrm{h}(s) \le 1$, $l + r \le 1/2 - 2\xi$, and $s \le 1/2$, the average is

$$\begin{aligned}\frac{n}{2} + \left((l + r + 2\varepsilon)(1-s) + \frac{\mathrm{h}(s)+s}{2}\right)\frac{n}{3} &\le \frac{n}{2} + \left(\frac{\mathrm{h}(s)+1}{2} - 2\xi(1-s)\right)\frac{n}{3} + \frac{2\varepsilon n}{3} \\ &\le \frac{n}{2} + \frac{(1-\xi)n}{3} + \frac{2\varepsilon n}{3}.\end{aligned}$$

□

Now fix a solution $\mathbf{c}$. For a given s, we call an integer k a *correct key* if it is the left key of some $\mathbf{a} \in \mathcal{A}$ with $\mathbf{c} - \mathbf{a} \in \mathcal{B}$. In this case, k is also the right key of $\mathbf{c} - \mathbf{a}$, since $\mathbf{c} \cdot \mathbf{z} = \boldsymbol{\tau}$.

Lemma 2.5. *For some $s \in \{0, 3/n, \ldots, 1/2\}$, there are at least $N_{\mathrm{sum}}/(n/3+2)$ distinct correct keys.*

Proof of Lemma 2.5. Each binary prefix $\mathbf{a}$ determines $\mathbf{b} \coloneqq \mathbf{c} - \mathbf{a}$ on the prefix. Complementing $\mathbf{a}$ changes the support size of this $\mathbf{b}$ from k to $n/3 - k$, and maps its prefix sum to the total prefix sum minus that sum. Consequently, at least $N_{\mathrm{sum}}/2$ distinct prefix sums have a representation with $k \le n/6$: every sum without one has a distinct complementary sum with one. Placing the outside part of $\mathbf{c}$ on the designated sides merely translates these sums by a fixed amount. Among the $n/6 + 1$ values of s, one therefore receives at least $N_{\mathrm{sum}}/(n/3+2)$ distinct correct keys. □

We call such a value of s the *good layer*.

Filtering. We filter the elements of $\mathcal{A}$ and $\mathcal{B}$ by their keys k, namely the left key for $\mathbf{a}$ and the right key for $\mathbf{b}$. Let p be the smallest power of two with $p > 2N_{\mathrm{sum}}(|\boldsymbol{\tau}| + \sum_i |z_i| + 1)$, and put

$$q \coloneqq \frac{p}{2^{\lfloor \log_2 N_{\mathrm{sum}} \rfloor}}.$$

Draw $h = h_{\alpha,\beta}$ uniformly from $\mathcal{H}_p$, and keep an element if and only if $h(\mathsf{k}) \in \mathcal{I}_q$ for its key k; we then say that the element and its key *survive*.

Lemma 2.6. *Every element of $\mathcal{A}$ and $\mathcal{B}$ survives with probability less than $2/N_{\mathrm{sum}}$. In the good layer, some correct key survives with probability at least $1/(n/3+3)$.*

Proof of Lemma 2.6. By Lemma 2.3, each key survives with probability $q/p = 2^{-\lfloor \log_2 N_{\mathrm{sum}} \rfloor} \in [1/N_{\mathrm{sum}}, 2/N_{\mathrm{sum}})$, which gives the first claim. Since $q \geq p/N_{\mathrm{sum}} > 2(|\boldsymbol{\tau}| + \sum_i |z_i| + 1)$, any two keys differ by at most $|\boldsymbol{\tau}| + 2\sum_i |z_i| < q$. By Lemma 2.3, the survival events of distinct keys are therefore pairwise independent.

Let X count the surviving distinct correct keys in the good layer. By Lemma 2.5, $\mathbb{E}[X] \geq 1/(n/3+2)$, and by pairwise independence, $\mathbb{E}[X^2] \leq \mathbb{E}[X] + (\mathbb{E}[X])^2$. Cauchy–Schwarz gives

$$\Pr[X > 0] \geq \frac{(\mathbb{E}[X])^2}{\mathbb{E}[X^2]} \geq \frac{1}{1 + 1/\mathbb{E}[X]} \geq \frac{1}{n/3+3}. \qquad \square$$

Next, we generate the surviving records without listing all of $\mathcal{A}$ and $\mathcal{B}$, using the meet-in-the-middle enumeration idea of [MNPW19, Lemma 3.8].

Lemma 2.7. *The surviving records can be listed in time $2^{o(n)}(2^{5n/12} + 2^{(1/2-\xi/6)n} + N_{\mathsf{L}} + N_{\mathsf{R}})$, where N_{L} and N_{R} are the numbers of surviving records in $\mathcal{A}$ and $\mathcal{B}$, respectively.*

Proof of Lemma 2.7. Split each of the three blocks into equal halves. Then every $\mathbf{a} \in \mathcal{A}$ is uniquely a sum $\mathbf{a}_1 + \mathbf{a}_2$ of partial assignments on the unions of the first and second halves of the three blocks, respectively, and we enumerate these into the *left half-lists* $\mathcal{A}_1$ and $\mathcal{A}_2$. Likewise $\mathbf{b} = \mathbf{b}_1 + \mathbf{b}_2$, giving the *right half-lists* $\mathcal{B}_1$ and $\mathcal{B}_2$; here we keep only half-prefixes with at most $sn/3$ nonzero entries, and group the elements of $\mathcal{B}_1$ by the number of nonzero entries in their half-prefixes.

Keys are additive over the halves. For $\mathbf{a} = \mathbf{a}_1 + \mathbf{a}_2$, put $\mathsf{k}_i \coloneqq \mathbf{a}_i \cdot \mathbf{z}$, so that $\mathbf{a}$ survives if and only if $\alpha\mathsf{k}_1 \bmod p$ lies in the cyclic interval $\mathcal{I}_q - \beta - \alpha\mathsf{k}_2$. So we sort $\mathcal{A}_1$ by $\alpha\mathsf{k}_1 \bmod p$, and for each $\mathbf{a}_2 \in \mathcal{A}_2$ report the entries of $\mathcal{A}_1$ in this interval. On the right, with $\mathsf{k}_i \coloneqq \mathbf{b}_i \cdot \mathbf{z}$, the interval is $\alpha\boldsymbol{\tau} + \beta - \alpha\mathsf{k}_2 - \mathcal{I}_q$; for each $\mathbf{b}_2 \in \mathcal{B}_2$, we search only the group of $\mathcal{B}_1$ whose members give exactly $sn/3$ nonzero entries in the prefix when added to $\mathbf{b}_2$. Each interval is a union of at most two ordinary intervals. Preserve all assignments, including equal residues.

This takes time proportional to the sizes of the four half-lists plus the reported records, times $2^{o(n)}$ for arithmetic and sorting. The left half-lists $\mathcal{A}_1, \mathcal{A}_2$ have at most $2^{5n/12+O(1)}$ entries each, since $\ell \leq n/2$. For either right half-list, the number of possible half-prefixes is at most

$$\sum_{j=0}^{sn/3} \binom{n/6}{j} 2^j \leq 2^{n/6+sn/3}.$$

Each of $\mathcal{B}_1, \mathcal{B}_2$ also has half of the $2n/3 - \ell$ binary variables in the second outside block, so substituting ℓ gives the size exponent

$$\begin{aligned}\frac{n}{6} + \frac{sn}{3} + \frac{1}{2}\left(\frac{2n}{3} - \ell\right) &= \frac{5n}{12} - \frac{\mathrm{h}(s)n}{12} + \frac{sn}{4} + \frac{(l-r)(1-s)n}{6} + O(1) \\ &\le \frac{n}{2} - \frac{\mathrm{h}(s)n}{12} + \frac{sn}{6} - \frac{\xi(1-s)n}{3} + O(1) \\ &\le \left(\frac{1}{2} - \frac{\xi}{6}\right)n + O(1),\end{aligned}$$

using $l - r \le 1/2 - 2\xi$ in the second step, and $\mathrm{h}(s) \ge 2s$ and $s \le 1/2$ in the last. □

Finally, we test the surviving records by half-stars partial match.

Lemma 2.8. *Given the surviving records, we can find a solution among the pairs of surviving records, or correctly report that none exists, in time* $2^{o(n)}\,\mathrm{cost}(N_{\mathsf{L}}, N_{\mathsf{R}})$.

Proof of Lemma 2.8. Sort the surviving records by their keys; since keys have $2^{o(n)}$ bits, this costs a factor of $2^{o(n)}$ per record. For each key value k occurring on both sides, let

$$\begin{aligned}\boldsymbol{A}_{\mathsf{k}} &\coloneqq \{\mathbf{x}(\mathbf{a}) : \mathbf{a} \text{ survives and has left key } \mathsf{k}\}, \\ \boldsymbol{B}_{\mathsf{k}} &\coloneqq \{\mathbf{y}(\mathbf{b}) : \mathbf{b} \text{ survives and has right key } \mathsf{k}\},\end{aligned}$$

and run the half-stars partial match algorithm on $(\boldsymbol{A}_{\mathsf{k}}, \boldsymbol{B}_{\mathsf{k}})$. By the translation to partial match, some call answers positively if and only if there are surviving $\mathbf{a}, \mathbf{b}$ with

$$\mathbf{a} \cdot \mathbf{z} = \boldsymbol{\tau} - \mathbf{b} \cdot \mathbf{z} \qquad \text{and} \qquad \mathbf{a} + \mathbf{b} \in \{0,1\}^n,$$

that is, if and only if some pair of surviving records forms a solution. The sets $\boldsymbol{A}_{\mathsf{k}}$ have total size at most N_{L}, and the sets $\boldsymbol{B}_{\mathsf{k}}$ total size at most N_{R}, so the total cost of these calls is $O(\mathrm{cost}(N_{\mathsf{L}}, N_{\mathsf{R}}))$.

If the call for k answers positively, repeatedly bisect $\boldsymbol{A}_{\mathsf{k}}$ and $\boldsymbol{B}_{\mathsf{k}}$, always retaining a half for which the answer remains positive. After $O(n)$ such calls, one matching pair $(\mathbf{x}(\mathbf{a}), \mathbf{y}(\mathbf{b}))$ remains, and $\mathbf{a} + \mathbf{b}$ is a solution. □

Putting things together. An *attempt* in layer s draws h, generates the surviving records by Lemma 2.7, and runs Lemma 2.8. Since $\mathrm{cost}(N_{\mathsf{L}}, N_{\mathsf{R}}) \ge N_{\mathsf{L}} + N_{\mathsf{R}}$, the last term of Lemma 2.7 is covered by Lemma 2.8. By Lemma 2.6 and linearity of expectation, $\mathbb{E}[N_{\mathsf{L}}] < 2|\mathcal{A}|/N_{\mathrm{sum}}$ and $\mathbb{E}[N_{\mathsf{R}}] < 2|\mathcal{B}|/N_{\mathrm{sum}}$, so filtering reduces the expected cost by a factor of at least $N_{\mathrm{sum}}/2$ compared to the unscreened cost. By Lemma 2.4, the expected time of one attempt is therefore at most

$$2^{o(n)}\left(2^{5n/12} + 2^{(1/2-\xi/6)n} + \frac{2^{n/2+(1-\xi)n/3+2\varepsilon n/3}}{N_{\mathrm{sum}}}\right).$$

This bound is known in advance, and we abort an attempt after n^2 times this bound; by Markov's inequality, this loses at most $1/n^2$ success probability.

If a solution exists, then by Lemmas 2.6 and 2.8, an attempt in the good layer finds a solution with probability at least $1/(n/3+3)$ if not aborted, hence before the cutoff with probability at least $1/(n/3+3) - 1/n^2 \ge 1/(2n)$. We perform n^2 independent attempts per layer, so the failure probability is exponentially small.

It remains to compare the two procedures. At the threshold $N_{\mathrm{sum}} = 2^{(1-2\xi/3)n/3}$, both principal costs are $2^{(1/2-\xi/9)n}$, because

$$\sqrt{\frac{N_{\mathrm{sum}}}{2^{n/3}}} = \frac{2^{(1-\xi)n/3}}{N_{\mathrm{sum}}} = 2^{-\xi n/9}.$$

The exponent of the right half-lists is smaller by $\xi/18$, and preprocessing and the left half-lists are also cheaper. Both procedures therefore run in time $2^{(1/2-\xi/9+2\varepsilon/3)n+o(n)}$. □

3 Depth-2 linear circuits

In this section, we obtain a nontrivial partial match algorithm by designing depth-2 linear circuits.

3.1 Preliminaries

We first fix our notation for depth-2 linear circuits, following [JS13, Section 1.2] and [AL25, Section 3.2]. A *depth-2 linear circuit* is a pair of matrices $C \coloneqq (\mathbf{L}_C, \mathbf{R}_C)$ whose product $\mathbf{L}_C\mathbf{R}_C$ is defined, and it computes the map $\mathbf{u} \mapsto \mathbf{u}\mathbf{L}_C\mathbf{R}_C$. The rows of $\mathbf{L}_C$ index the *input gates*, the columns of $\mathbf{L}_C$, equivalently the rows of $\mathbf{R}_C$, index the *middle gates*, and the columns of $\mathbf{R}_C$ index the *output gates*. We write $\mathrm{Mid}(C)$ for the set of middle gates. The nonzero entries of $\mathbf{L}_C$ and $\mathbf{R}_C$ are the *edges* of C, and their values are the *edge coefficients*.

We measure a circuit by its degrees. Write nnz for the number of nonzero entries. For an input gate $\mathbf{x}$ and an output gate $\mathbf{y}$, put

$$\deg_{\mathsf{L}}(C, \mathbf{x}) \coloneqq \mathrm{nnz}(\mathbf{L}_C[\mathbf{x}, *]), \qquad \deg_{\mathsf{R}}(C, \mathbf{y}) \coloneqq \mathrm{nnz}(\mathbf{R}_C[*, \mathbf{y}]).$$

Thus $\deg_{\mathsf{L}}(C, \mathbf{x})$ and $\deg_{\mathsf{R}}(C, \mathbf{y})$ count the edges at the gate. We write $\deg_{\mathsf{L}}(C)$ and $\deg_{\mathsf{R}}(C)$ for their maxima over all gates, and call $\mathrm{nnz}(\mathbf{L}_C) + \mathrm{nnz}(\mathbf{R}_C)$ the *size* of C.

Different requirements on the product $\mathbf{L}_C\mathbf{R}_C$ give different circuit models. All products are taken over the integers, and for a 01 matrix M we consider the following four models.

Model	Allowed coefficients	Requirement for computing a 01 matrix M
PAR	$0, 1$	$\mathbf{L}_C\mathbf{R}_C = M$
OR	$0, 1$	$(\mathbf{L}_C\mathbf{R}_C)_{\mathbf{x},\mathbf{y}} \neq 0 \iff M_{\mathbf{x},\mathbf{y}} = 1$
$\mathsf{EXACT}_{\mathbb{Z}}$	all integers	$\mathbf{L}_C\mathbf{R}_C = M$
$\mathsf{SUP}_{\mathbb{Z}}$	all integers	$(\mathbf{L}_C\mathbf{R}_C)_{\mathbf{x},\mathbf{y}} \neq 0 \iff M_{\mathbf{x},\mathbf{y}} = 1$

Here PAR stands for a partition of the ones of M into all-one rectangles. Our constructions are in the third model: C is an *exact integer circuit*, or $\mathsf{EXACT}_{\mathbb{Z}}$ circuit, for M if both matrices have integer entries and $\mathbf{L}_C\mathbf{R}_C = M$.

Next, we specialize to partial match. Recall from Section 2.1 that the rows of $P_1^{\otimes n}$ are indexed by binary strings and its columns by query strings. Accordingly, we name the input gates of a circuit for $P_1^{\otimes n}$ by binary strings $\mathbf{x}$ and its output gates by query strings $\mathbf{y}$; the string coordinates x_i should not be confused with the input values $u_{\mathbf{x}}$. The entries of $P_1^{\otimes n}$ are then polynomials in the coordinates of $\mathbf{x}$: for $\mathbf{y} \in \{0, 1, *\}^n$, put

$$f_{\mathbf{y}}(\mathbf{x}) \coloneqq \prod_{i:y_i=1} x_i \prod_{i:y_i=0} (1 - x_i),$$

so that $(P_1^{\otimes n})_{\mathbf{x},\mathbf{y}} = f_{\mathbf{y}}(\mathbf{x})$. Since half-stars partial match only involves queries with a fixed number of specified positions, we let $P_{n,s}$ be the submatrix of $P_1^{\otimes n}$ formed by the columns $\mathbf{y}$ with $|S(\mathbf{y})| = \lfloor sn \rfloor$, retaining all rows.

We measure circuits for $P_{n,s}$ by the growth rate of their degrees, in the same way as admissible exponent pairs. Let $\mathcal{K}(s)$ consist of the pairs $(l, r) \in \mathbb{R}^2_{\geq 0}$ for which there are $\mathsf{EXACT}_{\mathbb{Z}}$ circuits C_n computing $P_{n,s}$ with $\deg_{\mathsf{L}}(C_n) \leq 2^{(l+o(1))n}$ and $\deg_{\mathsf{R}}(C_n) \leq 2^{(r+o(1))n}$. Equivalently, for every $\varepsilon > 0$, every sufficiently large n has an $\mathsf{EXACT}_{\mathbb{Z}}$ circuit for $P_{n,s}$ with degrees at most $2^{(l+\varepsilon)n}$ and $2^{(r+\varepsilon)n}$. For $s = 1/2$, even lengths suffice: for odd n, appending 0 to both strings embeds $P_{n,1/2}$ into $P_{n+1,1/2}$ without increasing either degree.

Finally, we fix some notation for counting. For $0 \leq v \leq u$, put

$$\mathrm{h}(u, v) \coloneqq u \log_2 u - v \log_2 v - (u - v) \log_2 (u - v),$$

with $0 \log_2 0 \coloneqq 0$; thus $\mathrm{h}(t) = \mathrm{h}(1, t)$ is the binary entropy function of Section 2. For a string $\mathbf{x}$ and a symbol i, let $\#i(\mathbf{x})$ count the occurrences of i in $\mathbf{x}$.

3.2 From circuits to algorithms

For an algorithm, the mere existence of good circuits is not enough: we must be able to *explicitly construct* them, in the sense that the edges at any given input or output gate can be listed in time comparable to its degree. Lemma 3.1 reduces circuits for $P_{n,1/2}$ to tensor powers of a circuit for $P_{k,1/2}$. When k is fixed, the latter circuit can be hardcoded, and the construction becomes explicit for all n (Corollary 3.2) and hence partial match algorithms (Corollary 3.3).

We start with the combinatorial part, which does not yet care about explicitness. A circuit for $P_{k,1/2}$ only handles queries whose wildcards are spread evenly over blocks of length k, while a general query of $P_{n,1/2}$ may have its wildcards distributed arbitrarily. The following lemma shows that we can always permute the coordinates so that the wildcards become evenly spread, and that the price is paid only on the left degree.

Lemma 3.1. *Let $k, n \geq 2$ be even, and put $d \coloneqq \lceil n/k \rceil$. If E is an $\mathsf{EXACT}_{\mathbb{Z}}$ circuit computing $(P_{k,1/2})^{\otimes d}$, then there is an $\mathsf{EXACT}_{\mathbb{Z}}$ circuit C_n computing $P_{n,1/2}$ with*

$$\deg_{\mathsf{L}}(C_n) \leq \binom{kd}{d} \deg_{\mathsf{L}}(E), \qquad \deg_{\mathsf{R}}(C_n) \leq \deg_{\mathsf{R}}(E).$$

Proof of Lemma 3.1. Let $\mathbf{y}$ be a length-kd query with exactly $kd/2$ wildcards. We call a permutation of $[kd]$ *balancing* for $\mathbf{y}$ if, after dividing the permuted query into d consecutive blocks of length k, each block contains exactly $k/2$ wildcards.

Suppose that $\boldsymbol{\iota}$ is a family of permutations containing a balancing permutation for every such query. Given a length-n binary string $\mathbf{x}$, append $kd - n$ zeros; given a length-n query $\mathbf{y}$, append $(kd - n)/2$ zeros followed by $(kd - n)/2$ wildcards. The new positions all match, and the padded query has exactly $kd/2$ wildcards.

Construct C_n with middle gates $\{(e, \iota)\}_{e \in \mathrm{Mid}(E), \iota \in \boldsymbol{\iota}}$. Each input gate $\mathbf{x}$ uses every permutation ι of its padded binary string: copy the edges from that permuted string to e in E as edges $\mathbf{x} \to (e, \iota)$ in C_n, with the same coefficients. Each output gate $\mathbf{y}$ selects exactly one permutation $\iota(\mathbf{y}) \in \boldsymbol{\iota}$ balancing for its padded query: copy the edges from e to that permuted query in E as edges $(e, \iota(\mathbf{y})) \to \mathbf{y}$ in C_n, with the same coefficients. Padding and simultaneous coordinate permutations preserve matching, and only the selected copy contributes to the entry for $(\mathbf{x}, \mathbf{y})$. Thus C_n computes $P_{n,1/2}$, with $\deg_{\mathsf{L}}(C_n) \leq |\boldsymbol{\iota}| \deg_{\mathsf{L}}(E)$ and $\deg_{\mathsf{R}}(C_n) \leq \deg_{\mathsf{R}}(E)$.

It remains to construct $\boldsymbol{\iota}$ with $|\boldsymbol{\iota}| \le \binom{kd}{d}$, and to select a balancing permutation for each padded query. Given a length-kd query $\mathbf{y}$ with $|S(\mathbf{y})| = kd/2$, start with an empty stack and scan its coordinates $i = 1, \dots, kd$, pushing i onto the stack. Whenever the top k coordinates contain exactly $k/2$ wildcards in $\mathbf{y}$, remove them and map them, in increasing coordinate order, to the next block.

There are exactly d removals. At the end of each step, no consecutive k coordinates on the stack contain exactly $k/2$ wildcards: pushing can create only the top window, and removing it leaves a prefix of the previous stack. If the stack were nonempty at the end, its length would be divisible by k and exactly half its coordinates would contain wildcards. Divide it into blocks of length k; their wildcard counts have average $k/2$. A sliding window's wildcard count changes by at most 1 at each step, so some window must contain exactly $k/2$ wildcards, a contradiction. The stack is therefore empty at the end, defining a permutation $\iota(\mathbf{y})$.

The permutation $\iota(\mathbf{y})$ is determined by the d positions in $[kd]$ at which blocks are removed: simulating the stack procedure with these removal positions, ignoring the query characters, recovers it. Let $\boldsymbol{\iota}$ consist of the permutations obtained in this way from all d-subsets of $[kd]$ for which every removal can be performed. This family contains every $\iota(\mathbf{y})$ and has $|\boldsymbol{\iota}| \le \binom{kd}{d}$. □

The factor $\binom{kd}{d} \le 2^{\mathrm{h}(1/k)kd}$ has exponential rate $\mathrm{h}(1/k)$, which tends to 0 as k grows. So for any slack we can fix one good circuit C_0 for $P_{k,1/2}$ once and for all, and apply Lemma 3.1 to its tensor powers. Since the edges of a tensor power can be listed block by block, the resulting circuits are explicit.

Corollary 3.2. *Let $(l, r) \in \mathcal{K}(1/2)$ and $\varepsilon > 0$. There are $\mathsf{EXACT}_{\mathbb{Z}}$ circuits C_n computing $P_{n,1/2}$, one for each even $n \ge 2$, and deterministic algorithms A_{L} and A_{R} with the following properties.*

- *Given $\mathbf{x} \in \{0,1\}^n$, the algorithm A_{L} lists all edges $\mathbf{x} \to e$ of C_n, each as a middle-gate name e with its coefficient, in time $O\big(2^{(l+\varepsilon)n}\big)$.*
- *Given $\mathbf{y} \in \{0,1,*\}^n$ with $|S(\mathbf{y})| = n/2$, the algorithm A_{R} lists all edges $e \to \mathbf{y}$ of C_n, each as a middle-gate name e with its coefficient, in time $O\big(2^{(r+\varepsilon)n}\big)$.*

Proof of Corollary 3.2. Since $(l, r) \in \mathcal{K}(1/2)$, its definition lets us choose and fix an even $k \ge 2$ with $\mathrm{h}(1/k) \le \varepsilon/4$ and an $\mathsf{EXACT}_{\mathbb{Z}}$ circuit C_0 computing $P_{k,1/2}$ with $\deg_{\mathsf{L}}(C_0) \le 2^{(l+\varepsilon/4)k}$ and $\deg_{\mathsf{R}}(C_0) \le 2^{(r+\varepsilon/4)k}$. Hardcode this fixed circuit's adjacency lists and integer edge coefficients into the algorithms.

For each even $n \ge 2$, let $d := \lceil n/k \rceil$ and let C_n be the circuit obtained by applying Lemma 3.1 to $E := C_0^{\otimes d}$. We first list the edges of $C_0^{\otimes d}$ at a binary string or query of length kd: divide it into d blocks of length k, list the edges of each block in the hardcoded circuit C_0, and take the Cartesian product. This gives the names $e = (e_1, \dots, e_d)$, with edge coefficient equal to the product of the d block coefficients.

Next, recall from the proof of Lemma 3.1 that the middle gates of C_n are the pairs (e, ι), and that each $\iota \in \boldsymbol{\iota}$ is determined by its d removal positions; we name ι by the kd-bit indicator string of these positions, from which ι is recovered in time $n^{O(1)}$. Given $\mathbf{x}$, the algorithm A_{L} pads it, enumerates the at most $\binom{kd}{d}$ indicator strings, recovers each ι, and lists the edges of $C_0^{\otimes d}$ at the permuted padded string. Given $\mathbf{y}$, the algorithm A_{R} pads it, runs the stack procedure once to find $\iota(\mathbf{y})$, and lists the edges of $C_0^{\otimes d}$ at the permuted padded query. Names and edge coefficients have

$O(n)$ bits, since each coefficient is a product of d fixed integers. Therefore

$$\begin{aligned}\operatorname{time}(A_{\mathsf{L}}) &\le n^{O(1)}\binom{kd}{d}\deg_{\mathsf{L}}(C_0)^d \le n^{O(1)}2^{\mathrm{h}(1/k)kd}2^{(l+\varepsilon/4)kd}\\ &\le n^{O(1)}2^{(l+\varepsilon/2)kd} = O\Big(2^{(l+\varepsilon)n}\Big),\\ \operatorname{time}(A_{\mathsf{R}}) &\le n^{O(1)}\deg_{\mathsf{R}}(C_0)^d \le n^{O(1)}2^{(r+\varepsilon/4)kd} = O\Big(2^{(r+\varepsilon)n}\Big).\end{aligned}$$

Here k is fixed, so $kd = n + O(1)$, and the remaining exponent slack absorbs the polynomial factors and padding loss. This proves the corollary. □

Once the edges at every gate can be listed efficiently, we can turn these circuits into a partial match algorithm via "meet-in-the-middle".

Corollary 3.3. *Every $(l, r) \in \mathcal{K}(1/2)$ is an admissible exponent pair.*

Proof of Corollary 3.3. Use the algorithms $A_{\mathsf{L}}, A_{\mathsf{R}}$ from Corollary 3.2. For each $\mathbf{x} \in \boldsymbol{A}$, use A_{L} to enumerate all edges $\mathbf{x} \to e$ in C_n and add each edge coefficient to the left counter for e. For each $\mathbf{y} \in \boldsymbol{B}$, use A_{R} to enumerate all edges $e \to \mathbf{y}$ in C_n and add each edge coefficient to the right counter for e. For each e, multiply its left and right counters, taking absent counters to be zero, and sum the products. The result is

$$\sum_e \Big(\sum_{\mathbf{x}\in\boldsymbol{A}} (\mathbf{L}_{C_n})_{\mathbf{x},e}\Big)\Big(\sum_{\mathbf{y}\in\boldsymbol{B}} (\mathbf{R}_{C_n})_{e,\mathbf{y}}\Big) = \sum_{\mathbf{x}\in\boldsymbol{A},\,\mathbf{y}\in\boldsymbol{B}} (P_1^{\otimes n})_{\mathbf{x},\mathbf{y}}.$$

Since C_n computes $P_{n,1/2}$ over the integers, this is the number of matching pairs in $\boldsymbol{A}\times\boldsymbol{B}$. Output that a matching pair exists if and only if the result is positive.

Implement the counters by sorting the generated coefficients deterministically by e. Middle-gate names and edge coefficients have $O(n)$ bits, and $|\boldsymbol{A}| \le 2^n$, $|\boldsymbol{B}| \le 3^n$, so sorting and integer arithmetic introduce only a factor of $n^{O(1)}$. Using Corollary 3.2 with slack $\varepsilon/2$ gives the required running time. □

3.3 Colored volume and rebalancing

In this subsection we prove our degree version of rebalancing. This idea comes from previous works [JS13, AGP23, Ser22, AL25], where it was developed for optimizing circuit sizes; here we adapt the method to degrees.

A single circuit constructed in Section 3.4 does not improve the degrees on its own: its middle gates fall into groups with rather different left and right degrees. The idea of rebalancing is to take tensor powers, and to choose the circuit for the remaining tensor factors separately for each group, so that the two degrees even out.

To make this precise, we track the middle gates in groups given by a coloring. A coloring $\chi\colon \mathrm{Mid}(C) \to \mathcal{Z}$ assigns to every middle gate a color in the color set $\mathcal{Z}$; a circuit together with a coloring is a *colored circuit*. We usually suppress the coloring from the notation and simply speak of a colored circuit C. For $\zeta \in \mathcal{Z}$, let C_ζ retain all input and output gates and just the middle gates of color ζ. That is, $C_\zeta = (\mathbf{L}_\zeta, \mathbf{R}_\zeta)$, where $\mathbf{L}_\zeta$ consists of the columns of $\mathbf{L}_C$ and $\mathbf{R}_\zeta$ of the rows of $\mathbf{R}_C$ indexed by middle gates of color ζ, so that

$$\mathbf{L}_C\mathbf{R}_C = \sum_{\zeta\in\mathcal{Z}} \mathbf{L}_\zeta\mathbf{R}_\zeta.$$

3.3.1 Rebalancing

Now we sketch the basic idea of rebalancing. Let $C^{(1)}, \dots, C^{(m)}$ be m different colored circuits, and let $C^{(i)}_\zeta = (\mathbf{L}^{(i)}_\zeta, \mathbf{R}^{(i)}_\zeta)$ denote the ζ-colored part of $C^{(i)}$. Suppose that all of them compute the same desired matrix M, that is, the identity

$$M = \sum_{\zeta \in \mathcal{Z}} \mathbf{L}^{(i)}_\zeta \mathbf{R}^{(i)}_\zeta$$

holds for every i. For each single i, we can bound the degrees of $C^{(i)}$ by the modest estimate summing over the colors:

$$\deg_{\mathsf{L}}(C^{(i)}) \le \sum_{\zeta \in \mathcal{Z}} \deg_{\mathsf{L}}(C^{(i)}_\zeta), \qquad \deg_{\mathsf{R}}(C^{(i)}) \le \sum_{\zeta \in \mathcal{Z}} \deg_{\mathsf{R}}(C^{(i)}_\zeta). \tag{3.1}$$

Although this estimate is rarely tight, we will see later that it is more convenient for the rebalancing process.

Now consider the question of constructing circuits for the tensor power $M^{\otimes d}$ from the base circuits for M that we know. At first glance, the circuits we can build from the $C^{(i)}$ are the tensor products $C^{(i_1)} \otimes \cdots \otimes C^{(i_d)}$. The main insight of rebalancing is that we actually have a much richer family. Let us first consider the tensor square $M^{\otimes 2}$. We may first expand one tensor factor by some chosen circuit $C^{(i)}$, that is,

$$M^{\otimes 2} = \Big(\sum_{\zeta \in \mathcal{Z}} \mathbf{L}^{(i)}_\zeta \mathbf{R}^{(i)}_\zeta\Big) \otimes M = \sum_{\zeta \in \mathcal{Z}} \Big(\mathbf{L}^{(i)}_\zeta \mathbf{R}^{(i)}_\zeta\Big) \otimes M.$$

After pulling M inside the summation, notice that we actually have the freedom to choose a different circuit computing the inner M for each color ζ. Let $j \colon \mathcal{Z} \to [m]$ be such a choice; we can then write

$$M^{\otimes 2} = \sum_{\zeta \in \mathcal{Z}} \Big(\mathbf{L}^{(i)}_\zeta \mathbf{R}^{(i)}_\zeta\Big) \otimes \Big(\mathbf{L}_{C^{(j_\zeta)}} \mathbf{R}_{C^{(j_\zeta)}}\Big) = \sum_{\zeta \in \mathcal{Z}} \Big(\mathbf{L}^{(i)}_\zeta \otimes \mathbf{L}_{C^{(j_\zeta)}}\Big)\Big(\mathbf{R}^{(i)}_\zeta \otimes \mathbf{R}_{C^{(j_\zeta)}}\Big).$$

By the modest estimate (3.1), this circuit has

$$\deg_{\mathsf{L}} \le \sum_{\zeta \in \mathcal{Z}} \deg_{\mathsf{L}}(C^{(i)}_\zeta) \deg_{\mathsf{L}}(C^{(j_\zeta)}), \qquad \deg_{\mathsf{R}} \le \sum_{\zeta \in \mathcal{Z}} \deg_{\mathsf{R}}(C^{(i)}_\zeta) \deg_{\mathsf{R}}(C^{(j_\zeta)}). \tag{3.2}$$

In general, we can construct a family $\mathcal{F}_d$ of depth-2 linear circuits computing $M^{\otimes d}$ in the following recursive way.

First, $\mathcal{F}_0$ is a singleton, consisting of the trivial depth-2 linear circuit computing the 1-dimensional identity map, which has degree 1 on both sides.

Then for $d \ge 1$, we construct $\mathcal{F}_d$ from $\mathcal{F}_{d-1}$ as follows. Each circuit in $\mathcal{F}_d$ is given by first selecting an index $i \in [m]$ and a choice function $E \colon \mathcal{Z} \to \mathcal{F}_{d-1}$ that chooses a circuit E_ζ for the $(d-1)$-th power for every color ζ. The circuit is then given by the parts

$$M^{\otimes d} = \sum_{\zeta \in \mathcal{Z}} \Big(\mathbf{L}^{(i)}_\zeta \otimes \mathbf{L}_{E_\zeta}\Big)\Big(\mathbf{R}^{(i)}_\zeta \otimes \mathbf{R}_{E_\zeta}\Big). \tag{3.3}$$

We call the circuits in $\mathcal{F}_d$ the *rebalancings* of the base circuits $C^{(1)}, \dots, C^{(m)}$.

To find a degree bound for the best circuit in $\mathcal{F}_d$, our strategy is to optimize a *weighted* degree instead. For a weight $\lambda > 0$, define

$$F^\star_d(\lambda) \coloneqq \min_{E \in \mathcal{F}_d} \{\deg_{\mathsf{L}}(E) + \lambda \deg_{\mathsf{R}}(E)\}.$$

This way we no longer need to control the two profiles $\deg_{\mathsf{L}}$ and $\deg_{\mathsf{R}}$ simultaneously, which is hard to keep track of when taking tensor products. Write $a_\zeta^{(i)} := \deg_{\mathsf{L}}(C_\zeta^{(i)})$ and $b_\zeta^{(i)} := \deg_{\mathsf{R}}(C_\zeta^{(i)})$; after discarding middle gates without edges, these are positive. For the circuit E' constructed by (3.3), the same relaxation as in (3.2) gives

$$\begin{aligned}\deg_{\mathsf{L}}(E') + \lambda \deg_{\mathsf{R}}(E') &\le \sum_{\zeta\in\mathcal{Z}} (a_\zeta^{(i)} \deg_{\mathsf{L}}(E_\zeta) + \lambda b_\zeta^{(i)} \deg_{\mathsf{R}}(E_\zeta)) \\ &= \sum_{\zeta\in\mathcal{Z}} a_\zeta^{(i)} \left(\deg_{\mathsf{L}}(E_\zeta) + \lambda \frac{b_\zeta^{(i)}}{a_\zeta^{(i)}} \deg_{\mathsf{R}}(E_\zeta) \right).\end{aligned}$$

Now suppose we want to find the best E', which comes from first determining some $i \in [m]$ and then the choice $E\colon \mathcal{Z} \to \mathcal{F}_{d-1}$. From the form above, the choice actually splits: each E_ζ can be optimized independently, with the rescaled weight $\lambda b_\zeta^{(i)}/a_\zeta^{(i)}$. Choosing every E_ζ optimally, we obtain a circuit E' with

$$\deg_{\mathsf{L}}(E') + \lambda \deg_{\mathsf{R}}(E') \le G_d^{(i)}(\lambda) := \sum_{\zeta\in\mathcal{Z}} a_\zeta^{(i)} F_{d-1}^\star \left(\lambda \frac{b_\zeta^{(i)}}{a_\zeta^{(i)}} \right).$$

So $G_d^{(i)}(\lambda)$ is the best bound we get when the outermost circuit is $C^{(i)}$. Since the family $\mathcal{F}_d$ goes through all $C^{(i)}$, we have

$$F_d^\star(\lambda) \le \min_{i\in[m]} G_d^{(i)}(\lambda).$$

3.3.2 Colored volume

The value of $G_d^{(i)}(\lambda)$ depends on the values of $F_{d-1}^\star$ at other points, which makes this recursion seem hard to analyze. However, the behaviour of this functional recursion has been resolved in the work of Alman and Li [AL25], which is the key input of this section.

We begin with some notation before stating their result. Notice that the recursion above depends only on the degrees of the color classes, and no longer on the specific details of the circuits. This allows the following simplification. A *degree-pair list* is a finite nonempty list W of pairs (a, b) of reals with $a, b \ge 1$; its *volume* is $\rho_W(\tau) := \sum_{(a,b)\in W} a^\tau b^{1-\tau}$ for $\tau \in [0, 1]$. We call the one-element lists $[(A, 1)]$ and $[(1, B)]$ with $A, B \ge 1$ *trivial*.

Lemma 3.4 (Essentially [AL25, Theorem B.2])**.** *Let $W_1, \dots, W_m$ be degree-pair lists, including a trivial list of each of the two forms. Define functions $F_d\colon (0,\infty) \to (0,\infty)$ by the recursion*

$$F_0(\lambda) := 1 + \lambda, \qquad F_d(\lambda) := \min_{1\le i\le m} \sum_{(a,b)\in W_i} a\, F_{d-1}\left(\lambda \frac{b}{a}\right) \qquad (d \ge 1).$$

Then we have the estimate

$$F_d(\lambda) \le 2^{o(d)} \max_{0\le\tau\le 1} \rho(\tau)^d \lambda^{1-\tau},$$

where the $o(d)$ term depends on $W_1, \dots, W_m$ but not on λ, and ρ is defined by

$$\rho(\tau) := \min_{1\le i\le m} \rho_{W_i}(\tau).$$

We remark that the original statement in [AL25, Appendix B], building on [JS13, AGP23, Ser22], is more concrete: it is stated for optimizing the size of circuits. However, a look at the proof shows that it makes no use of special properties of circuit sizes, so it applies verbatim to abstract degree-pair lists as well.

To connect the lemma with the sketch above, take $W_i \coloneqq [(a_\zeta^{(i)}, b_\zeta^{(i)})]_{\zeta\in\mathcal{Z}}$. Since $F_0^\star(\lambda) = 1+\lambda$ and the right-hand side of the recursion is monotone in F_{d-1}, induction on d shows $F_d^\star \le F_d$. Thus the lemma bounds the weighted degree of the best rebalancing in terms of the volumes $\rho_{W_i}(\tau)$ alone. This motivates us to consider the *colored volume* of circuits, which is exactly the volume of this degree-pair list. Let $\tau \in [0,1]$ be a parameter. The *τ-colored volume* of a colored circuit C is defined by

$$\rho_C(\tau) \coloneqq \sum_{\zeta\in\mathcal{Z}} \deg_{\mathsf{L}}(C_\zeta)^\tau \deg_{\mathsf{R}}(C_\zeta)^{1-\tau}.$$

Thus $\rho_C(1)$ and $\rho_C(0)$ are the sums of the left and of the right degrees of the color classes, and $\rho_C(\tau)$ interpolates between them. For a single color, $\rho_C(\tau) = \deg_{\mathsf{L}}(C)^\tau \deg_{\mathsf{R}}(C)^{1-\tau}$.

We now turn Lemma 3.4 back into a statement about circuits, which makes the sketch above precise.

Corollary 3.5. *Let $C^{(1)},\dots,C^{(m)}$ be colored $\mathsf{EXACT}_{\mathbb{Z}}$ circuits computing the same nonzero matrix M, among which there are a single-colored circuit with right degree 1 and a single-colored circuit with left degree 1, and put*

$$\rho(\tau) \coloneqq \min_{1\le i\le m} \rho_{C^{(i)}}(\tau).$$

Then for every $d \ge 0$ and every $\lambda > 0$ there is an $\mathsf{EXACT}_{\mathbb{Z}}$ circuit $E_{d,\lambda}$ computing $M^{\otimes d}$ with

$$\deg_{\mathsf{L}}(E_{d,\lambda}) + \lambda \deg_{\mathsf{R}}(E_{d,\lambda}) \le 2^{o(d)} \max_{0\le\tau\le1} \rho(\tau)^d \lambda^{1-\tau},$$

where the $o(d)$ term depends on $C^{(1)},\dots,C^{(m)}$ but not on λ.

For example, the trivial circuits (M, I) and (I, M), each with a single color, can always be added to the list.

Proof of Corollary 3.5. For each i, first delete middle gates with no incoming or no outgoing edge; this changes neither the computed matrix nor increases any degree or colored volume. Let $\mathcal{Z}_i$ be the nonempty colors of $C^{(i)}$, and define $a_\zeta^{(i)} \coloneqq \deg_{\mathsf{L}}(C_\zeta^{(i)})$ and $b_\zeta^{(i)} \coloneqq \deg_{\mathsf{R}}(C_\zeta^{(i)})$. These degrees are positive integers, so $W_i \coloneqq [(a_\zeta^{(i)}, b_\zeta^{(i)})]_{\zeta\in\mathcal{Z}_i}$ is a degree-pair list with $\rho_{W_i}(\tau) = \rho_{C^{(i)}}(\tau)$ for all $\tau \in [0,1]$. Since $M \ne 0$, the two single-colored circuits of the assumption give trivial lists of both forms. Let F_d be as in Lemma 3.4 for $W_1,\dots,W_m$; then its function ρ is the one above, and it remains to construct, for every $d \ge 0$ and $\lambda > 0$, a circuit $E_{d,\lambda}$ computing $M^{\otimes d}$ with $\deg_{\mathsf{L}}(E_{d,\lambda}) + \lambda \deg_{\mathsf{R}}(E_{d,\lambda}) \le F_d(\lambda)$.

The recursion for F_d tells us how to build such circuits: at each level, pick the best base circuit, and recurse on each of its color classes with the weight λ rescaled by the ratio of the two degrees. We proceed by induction on d. At $d = 0$, take a circuit with one middle gate and two edge coefficients equal to 1. For $d \ge 1$, choose at the root an index i attaining the minimum in $F_d(\lambda)$. For every color $\zeta \in \mathcal{Z}_i$, apply the induction hypothesis at $\lambda b_\zeta^{(i)}/a_\zeta^{(i)}$ to obtain $E_\zeta \coloneqq E_{d-1,\lambda b_\zeta^{(i)}/a_\zeta^{(i)}}$, and set

$$E_{d,\lambda} \coloneqq \sum_{\zeta\in\mathcal{Z}_i} C_\zeta^{(i)} \otimes E_\zeta,$$

as in (3.3). This sum shares the input and output gates and combines the branches' disjoint middle gates. A middle gate in each tensor product has name (e, f), where e and f name middle gates in its two factors; each edge coefficient is the product of the corresponding integer coefficients. The matrices computed by the $C_\zeta^{(i)}$ sum to M, and each E_ζ computes $M^{\otimes(d-1)}$. Hence $E_{d,\lambda}$ computes $M^{\otimes d}$. Tensor products multiply the two degrees, and combining the branches gives each degree at most the sum of the branch degrees. The induction hypothesis therefore bounds $\deg_{\mathsf{L}}(E_{d,\lambda}) + \lambda \deg_{\mathsf{R}}(E_{d,\lambda})$ by $\sum_{\zeta \in \mathcal{Z}_i} a_\zeta^{(i)} F_{d-1}(\lambda b_\zeta^{(i)}/a_\zeta^{(i)}) = F_d(\lambda)$. □

The following lemma is the form in which we use rebalancing: a bound on the colored volume of circuits for $P_{k,1/2}$ yields points of $\mathcal{K}(1/2)$.

Lemma 3.6. *Let $V\colon [0,1] \to \mathbb{R}_{\geq 0}$, and suppose that for every even $k \geq 2$ there is a colored $\mathsf{EXACT}_{\mathbb{Z}}$ circuit C_k computing $P_{k,1/2}$ such that*

$$\rho_{C_k}(\tau) \leq 2^{(V(\tau)+o(1))k} \qquad \text{uniformly in } \tau \in [0,1].$$

Then every pair $(l, r) \in \mathbb{R}^2$ with

$$\tau l + (1-\tau) r \geq \min\left\{V(\tau), \tau, \frac{1-\tau}{2}\right\} \qquad \text{for all } \tau \in [0,1]$$

belongs to $\mathcal{K}(1/2)$.

In the condition on (l, r), the terms τ and $(1-\tau)/2$ come from the two trivial circuits $(P_{k,1/2}, I)$ and $(I, P_{k,1/2})$, with degrees $(\binom{k}{k/2}, 1)$ and $(1, 2^{k/2})$; rebalancing lets us use whichever of the three circuits is best at each τ.

Proof of Lemma 3.6. The plan is as follows. We apply Corollary 3.5 to the given circuit and the two trivial circuits to obtain circuits for the tensor powers $(P_{k,1/2})^{\otimes d}$, and then pass to $P_{n,1/2}$ by Lemma 3.1.

The assumptions at $\tau = 1$ and $\tau = 0$ give $l, r \geq 0$. Fix $\varepsilon > 0$. By the definition of $\mathcal{K}(1/2)$, it suffices to construct, for every sufficiently large n, a circuit computing $P_{n,1/2}$ with degrees $\leq 2^{(l+\varepsilon)n}$ and $\leq 2^{(r+\varepsilon)n}$. By the hypothesis, we can choose and fix an even k with $\mathrm{h}(1/k) \leq \varepsilon/4$ such that $C^{(1)} := C_k$ satisfies $\rho_{C^{(1)}}(\tau) \leq 2^{(V(\tau)+\varepsilon/4)k}$ for every $\tau \in [0,1]$.

Add the two trivial circuits $C^{(2)} := (P_{k,1/2}, I)$ and $C^{(3)} := (I, P_{k,1/2})$, each colored with a single color. Their colored volumes are $\rho_{C^{(2)}}(\tau) = \binom{k}{k/2}^{\tau} \leq 2^{\tau k}$ and $\rho_{C^{(3)}}(\tau) = 2^{(1-\tau)k/2}$. Together with the assumption on (l, r), this gives $\rho(\tau) \leq 2^{(\tau l + (1-\tau) r + \varepsilon/4)k}$ for the function ρ of Corollary 3.5.

For every even $n \geq 4$, put $d := \lceil n/k \rceil$ and $\lambda := 2^{(l-r)kd}$. Then $\rho(\tau)^d \lambda^{1-\tau} \leq 2^{(l+\varepsilon/4)kd}$ for every τ, so Corollary 3.5 gives a circuit $E_{d,\lambda}$ computing $(P_{k,1/2})^{\otimes d}$ with $\deg_{\mathsf{L}}(E_{d,\lambda}) \leq 2^{(l+\varepsilon/4)kd+o(d)}$ and $\deg_{\mathsf{R}}(E_{d,\lambda}) \leq \lambda^{-1} 2^{(l+\varepsilon/4)kd+o(d)} = 2^{(r+\varepsilon/4)kd+o(d)}$. Applying Lemma 3.1 to $E_{d,\lambda}$ gives a circuit C_n computing $P_{n,1/2}$ with $\deg_{\mathsf{L}}(C_n) \leq 2^{(l+\varepsilon/4+\mathrm{h}(1/k))kd+o(d)}$ and $\deg_{\mathsf{R}}(C_n) \leq 2^{(r+\varepsilon/4)kd+o(d)}$.

Since k is fixed, $kd - n < k$ and $o(d) = o(n)$. Choose an integer $n_0 \geq 4$ so that, for every integer $n \geq n_0$ and $d = \lceil n/k \rceil$,

$$o(d) + (\max\{l, r\} + \varepsilon)(kd - n) \leq \frac{\varepsilon n}{4}.$$

For every even $n \geq n_0$, the two degree upper bounds are then $2^{(l+3\varepsilon/4)n}$ and $2^{(r+\varepsilon/2)n}$. For odd n, use the embedding into length $n+1$ from Section 3.1. Because k is even, $\lceil (n+1)/k \rceil = \lceil n/k \rceil$, so the same degree upper bounds in terms of kd and the same threshold condition apply. Since $\varepsilon > 0$ was arbitrary, $(l, r) \in \mathcal{K}(1/2)$. □

3.4 The circuit construction

In this subsection, we construct an $\mathsf{EXACT}_{\mathbb{Z}}$ circuit achieving $l + r < 1/2$.

We first construct a circuit for a structured set of columns, and then use the symmetries of $P_{k,1/2}$ and a random covering to handle all queries. Throughout this subsection, k is even and $a, b > 0$ are such that ak and bk are integers with $4ak \le k - bk$; in the proof of Theorem 3.8 we reduce to this case by rounding.

All exponents below come from counting with binomial coefficients. Recall the function $\mathrm{h}(u, v) = u \log_2 u - v \log_2 v - (u - v) \log_2(u - v)$ from Section 3.1. Since $\mathrm{h}(u, v) = u\,\mathrm{h}(v/u)$, the standard estimate $2^{n\,\mathrm{h}(m/n)}/(n+1) \le \binom{n}{m} \le 2^{n\,\mathrm{h}(m/n)}$ with $n = uk$ and $m = vk$ becomes

$$\frac{2^{\mathrm{h}(u,v)k}}{uk+1} \le \binom{uk}{vk} \le 2^{\mathrm{h}(u,v)k} \qquad \text{whenever } uk, vk \text{ are integers.} \tag{3.4}$$

In particular, $\binom{uk}{vk} = 2^{\mathrm{h}(u,v)k + O(\log k)}$ and $\binom{2m}{m} \ge 2^{2m}/(2m+1)$.

For $B \subseteq [k]$, let $\mathcal{Y}_B$ consist of the queries $\mathbf{y}$ with $|S(\mathbf{y})| = k/2$ whose specified bits outside B are balanced: $S(\mathbf{y}) \setminus B$ contains exactly ak positions i with $y_i = 1$ and exactly ak positions i with $y_i = 0$.

Lemma 3.7. *Let $B \subseteq [k]$ with $|B| = bk$. Then the restriction of $P_{k,1/2}$ to the columns $\mathcal{Y}_B$ has a colored $\mathsf{EXACT}_{\mathbb{Z}}$ circuit C_B, whose colors are ordered pairs (μ, θ) satisfying $a \le \mu \le 2a$, $\mu \le \theta \le 2\mu$, and $\mu k, \theta k \in \mathbb{Z}$.*

Moreover, for every color (μ, θ) and every binary string $\mathbf{x}$, writing $C' \coloneqq (C_B)_{(\mu,\theta)}$, we have

$$\begin{aligned} \deg_{\mathsf{L}}(C') &\le O^*\big(2^{(\mathrm{h}(1-b,\mu)-2\mu)k}\big), \\ \mathbb{E}_{\mathbf{z}}\big[\deg_{\mathsf{L}}(C', \mathbf{x} \oplus \mathbf{z})\big] &\le O^*\big(2^{(\mathrm{h}(1-b,\mu)+\mathrm{h}(2\mu,\theta)-4\mu)k}\big), \\ \deg_{\mathsf{R}}(C') &\le 2^{(b-1/2+2a+\mathrm{h}(a,\mu-a)+\mathrm{h}(\mu,\theta-\mu))k}, \end{aligned}$$

where $\mathbf{z} \in \{0,1\}^k$ is uniformly random subject to $z_i = 0$ for all $i \in B$, and $\oplus$ is the bitwise XOR. Here $O^*(\cdot)$ hides only factors polynomial in k, uniformly in B and the color.

Proof of Lemma 3.7. The circuit treats the positions inside and outside B differently. Inside B, it simply enumerates the assignments of $\mathbf{x}$. Outside B, it expands the constraints by inclusion–exclusion and evaluates each resulting monomial using a covering family.

For $\mathbf{y} \in \mathcal{Y}_B$ and $i \in \{0,1\}$, let $A_i(\mathbf{y}) \coloneqq \{j \in S(\mathbf{y}) \setminus B : y_j = i\}$, so that $|A_0(\mathbf{y})| = |A_1(\mathbf{y})| = ak$. Expanding $\prod_{j \in A_0(\mathbf{y})}(1 - x_j)$ gives

$$f_{\mathbf{y}}(\mathbf{x}) = f_{\mathbf{y}|_B}(\mathbf{x}|_B) \sum_{A_1(\mathbf{y}) \subseteq U \subseteq A_1(\mathbf{y}) \cup A_0(\mathbf{y})} (-1)^{|U| - ak} \prod_{i \in U} x_i.$$

Every set U in this sum satisfies $U \subseteq \overline{B}$ and $ak \le |U| \le 2ak$. For every integer μk in this range, $2\mu k \le 4ak \le k - bk$, we let $\mathcal{Q}_{\overline{B},\mu}$ be a family of $2\mu k$-subsets of $\overline{B}$ that covers all μk-subsets of $\overline{B}$. By a classical probabilistic argument, such a family exists with

$$|\mathcal{Q}_{\overline{B},\mu}| \le k \frac{\binom{k-bk}{\mu k}}{\binom{2\mu k}{\mu k}}.$$

Indeed, a uniformly random $2\mu k$-subset of $\overline{B}$ contains a fixed μk-subset with probability $\binom{2\mu k}{\mu k}/\binom{k-bk}{\mu k}$, so with this many independent random sets, each of the at most 2^k subsets is

missed with probability at most e^{-k}, and a union bound applies. By (3.4), $\binom{k-bk}{\mu k} \le 2^{\mathrm{h}(1-b,\mu)k}$, and $\binom{2\mu k}{\mu k} \ge 2^{2\mu k}/(2\mu k+1)$, so

$$|\boldsymbol{Q}_{\overline{B},\mu}| \le k(2\mu k+1)\, 2^{(\mathrm{h}(1-b,\mu)-2\mu)k}.$$

Assign each μk-subset $U \subseteq \overline{B}$ to exactly one containing set $Q(U) \in \boldsymbol{Q}_{\overline{B},\mu}$.

The middle gates have names $(Q,\boldsymbol{\sigma})$, where $Q \in \boldsymbol{Q}_{\overline{B},\mu}$ for some μ and $\boldsymbol{\sigma} \in \{0,1\}^{B\cup Q}$ is an assignment to the positions in $B \cup Q$. Set

$$(\mathbf{L}_{C_B})_{\mathbf{x},(Q,\boldsymbol{\sigma})} \coloneqq \mathbf{1}[\mathbf{x}|_{B\cup Q} = \boldsymbol{\sigma}],$$
$$(\mathbf{R}_{C_B})_{(Q,\boldsymbol{\sigma}),\mathbf{y}} \coloneqq f_{\mathbf{y}|_B}(\boldsymbol{\sigma}|_B) \sum_{\substack{Q(U)=Q \\ A_1(\mathbf{y})\subseteq U\subseteq A_1(\mathbf{y})\cup A_0(\mathbf{y})}} (-1)^{|U|-ak} \prod_{i\in U} \sigma_i.$$

For each Q, the binary string $\mathbf{x}$ is adjacent to exactly one middle gate $(Q,\mathbf{x}|_{B\cup Q})$. Since each U is assigned to exactly one Q, the expansion above gives $(\mathbf{L}_{C_B}\mathbf{R}_{C_B})_{\mathbf{x},\mathbf{y}} = f_{\mathbf{y}}(\mathbf{x})$ over the integers for every binary string $\mathbf{x}$ and every $\mathbf{y} \in \mathcal{Y}_B$.

Give the middle gate $(Q,\boldsymbol{\sigma})$ the color (μ,θ), where $\mu \coloneqq |Q|/(2k)$ and $\theta \coloneqq \#1(\boldsymbol{\sigma}|_Q)/k$, and delete middle gates with no incoming or no outgoing edge. A nonzero right coefficient requires some $U \subseteq Q$ with $|U| = \mu k$ and $\boldsymbol{\sigma}|_U \equiv 1$, so the remaining colors satisfy $\mu \le \theta \le 2\mu$.

For the left degree, the middle gates of color (μ,θ) adjacent to $\mathbf{x}$ are among the gates $(Q,\mathbf{x}|_{B\cup Q})$ with $Q \in \boldsymbol{Q}_{\overline{B},\mu}$ and $\#1(\mathbf{x}|_Q) = \theta k$. Hence $\deg_{\mathsf{L}}(C',\mathbf{x}) \le |\boldsymbol{Q}_{\overline{B},\mu}|$, which gives the first bound. For the second, every $Q \in \boldsymbol{Q}_{\overline{B},\mu}$ lies in $\overline{B}$, so $(\mathbf{x}\oplus\mathbf{z})|_Q$ is uniform on $\{0,1\}^Q$, and

$$\mathbb{E}_{\mathbf{z}}[\deg_{\mathsf{L}}(C',\mathbf{x}\oplus\mathbf{z})] \le |\boldsymbol{Q}_{\overline{B},\mu}|\, 2^{-2\mu k}\binom{2\mu k}{\theta k} \le |\boldsymbol{Q}_{\overline{B},\mu}|\, 2^{(\mathrm{h}(2\mu,\theta)-2\mu)k},$$

using $\binom{2\mu k}{\theta k} \le 2^{\mathrm{h}(2\mu,\theta)k}$ from (3.4). Together with the bound on $|\boldsymbol{Q}_{\overline{B},\mu}|$, the exponent is $\mathrm{h}(1-b,\mu) - 2\mu + \mathrm{h}(2\mu,\theta) - 2\mu$, which gives the second bound.

For the right degree, fix $\mathbf{y} \in \mathcal{Y}_B$. A nonzero right coefficient requires $\boldsymbol{\sigma}|_B$ to agree with $\mathbf{y}$ on $S(\mathbf{y}) \cap B$, which has $k/2 - 2ak$ positions; this leaves $2^{bk-k/2+2ak}$ assignments on B. It also requires some U with $Q(U) = Q$ and $\boldsymbol{\sigma}|_U \equiv 1$. There are $\binom{ak}{\mu k-ak}$ choices of U of size μk, each of which determines Q and $\boldsymbol{\sigma}|_U$. Then $\binom{\mu k}{\theta k-\mu k}$ assignments on $Q \setminus U$ give the color (μ,θ). Counting a middle gate more than once gives the upper bound

$$2^{bk-k/2+2ak}\binom{ak}{\mu k-ak}\binom{\mu k}{\theta k-\mu k} \le 2^{(b-1/2+2a+\mathrm{h}(a,\mu-a)+\mathrm{h}(\mu,\theta-\mu))k},$$

where the three factors contribute the exponents $b-1/2+2a$, $\mathrm{h}(a,\mu-a)$, and $\mathrm{h}(\mu,\theta-\mu)$, respectively, by (3.4). □

We now assemble the circuits of Lemma 3.7 into a single colored circuit for the whole matrix $P_{k,1/2}$: flipping bits reaches the unbalanced queries, and a random family of pairs $(B,\boldsymbol{\varphi})$ covers all of them. The following theorem bounds the two degrees of every color class of the resulting circuit.

Theorem 3.8. *Let $a,b \in \mathbb{R}$ satisfy $a > 0$ and $4a \le 1-b \le 1/2$, and put*

$$\kappa \coloneqq \mathrm{h}(b) - \mathrm{h}(1/2,2a) - \mathrm{h}(1/2,1-b-2a).$$

Assume that $\kappa > 4a$. For $a \le \mu \le 2a$ and $\mu \le \theta \le 2\mu$, define

$$V_{\mathsf{L}}(\mu,\theta) \coloneqq \kappa + \mathrm{h}(1-b,\mu) + \mathrm{h}(2\mu,\theta) - 4\mu,$$
$$V_{\mathsf{R}}(\mu,\theta) \coloneqq b - \tfrac{1}{2} + 2a + \mathrm{h}(a,\mu-a) + \mathrm{h}(\mu,\theta-\mu).$$

Then for every sufficiently large even k, there is a colored $\mathsf{EXACT}_{\mathbb{Z}}$ circuit C_k computing $P_{k,1/2}$, with at most $(k+1)^2$ colors, each of which is a pair (μ,θ) with $a \le \mu \le 2a$ and $\mu \le \theta \le 2\mu$.

Moreover, for every color (μ,θ), writing $C' \coloneqq (C_k)_{(\mu,\theta)}$, we have

$$\deg_{\mathsf{L}}(C') \le 2^{(V_{\mathsf{L}}(\mu,\theta)+o(1))k}, \qquad \deg_{\mathsf{R}}(C') \le 2^{(V_{\mathsf{R}}(\mu,\theta)+o(1))k},$$

where the $o(1)$ terms are uniform over all colors.

Proof of Theorem 3.8. We construct the required colored circuits for every sufficiently large even k. To simplify notation, we write ak and bk for $\lfloor ak \rfloor$ and $\lfloor bk \rfloor$ throughout the proof; that is, we argue as if ak and bk were integers. The effect of this rounding is handled in the last paragraph.

Flipping symmetry. Lemma 3.7 handles only the queries whose specified bits outside B are balanced: ak ones and ak zeros. To reach the other queries, we use the symmetry of P_1 that swaps the roles of 0 and 1. For $\boldsymbol{\varphi} \in \{0,1\}^k$, write $\mathbf{x} \oplus \boldsymbol{\varphi}$ for the bitwise XOR, and let $\mathbf{y} \oplus \boldsymbol{\varphi}$ be the query obtained from $\mathbf{y}$ by replacing y_i with $y_i \oplus \varphi_i$ for every $i \in S(\mathbf{y})$ and keeping the wildcards. Then $f_{\mathbf{y}\oplus\boldsymbol{\varphi}}(\mathbf{x} \oplus \boldsymbol{\varphi}) = f_{\mathbf{y}}(\mathbf{x})$, and $|S(\mathbf{y} \oplus \boldsymbol{\varphi})| = |S(\mathbf{y})|$.

For a bk-subset B and a vector $\boldsymbol{\varphi} \in \{0,1\}^{\overline{B}}$, extended by zeros on B, let $\mathcal{Y}_{B,\boldsymbol{\varphi}} \coloneqq \{\mathbf{y} : \mathbf{y} \oplus \boldsymbol{\varphi} \in \mathcal{Y}_B\}$. Let C_B be the circuit of Lemma 3.7, and define $C_{B,\boldsymbol{\varphi}}$ by relabeling its input and output gates:

$$(\mathbf{L}_{C_{B,\boldsymbol{\varphi}}})_{\mathbf{x},e} \coloneqq (\mathbf{L}_{C_B})_{\mathbf{x}\oplus\boldsymbol{\varphi},e}, \qquad (\mathbf{R}_{C_{B,\boldsymbol{\varphi}}})_{e,\mathbf{y}} \coloneqq (\mathbf{R}_{C_B})_{e,\mathbf{y}\oplus\boldsymbol{\varphi}} \qquad (\mathbf{y} \in \mathcal{Y}_{B,\boldsymbol{\varphi}}),$$

with the same coloring. By the symmetry, $C_{B,\boldsymbol{\varphi}}$ is an $\mathsf{EXACT}_{\mathbb{Z}}$ circuit computing the restriction of $P_{k,1/2}$ to the columns $\mathcal{Y}_{B,\boldsymbol{\varphi}}$. It has the same right degrees as C_B, and the left degree of $\mathbf{x}$ in $C_{B,\boldsymbol{\varphi}}$ equals the left degree of $\mathbf{x} \oplus \boldsymbol{\varphi}$ in C_B, color by color.

Covering all queries. We choose a family $\mathcal{B}$ of pairs $(B,\boldsymbol{\varphi})$ such that the sets $\mathcal{Y}_{B,\boldsymbol{\varphi}}$ cover all queries. Choose B uniformly among the bk-subsets of $[k]$, and then choose the bits of $\boldsymbol{\varphi} \in \{0,1\}^{\overline{B}}$ independently and fairly. A fixed query $\mathbf{y}$ belongs to $\mathcal{Y}_{B,\boldsymbol{\varphi}}$ if and only if $\overline{B}$ contains exactly $2ak$ specified positions of $\mathbf{y}$ and the flipped bits $y_i \oplus \varphi_i$ on these positions contain exactly ak ones. This happens with probability

$$p_0 \coloneqq \frac{\binom{k/2}{2ak}\binom{k/2}{k-bk-2ak}}{\binom{k}{bk}} \frac{\binom{2ak}{ak}}{2^{2ak}},$$

which is the same for all queries. Here the first factor is the hypergeometric probability that exactly $2ak$ of the $k/2$ specified positions of $\mathbf{y}$ fall into the $(k-bk)$-set $\overline{B}$, and the second is the probability that $2ak$ fair bits contain exactly ak ones. We estimate each binomial coefficient by (3.4):

$$\binom{k/2}{2ak} = 2^{\mathrm{h}(1/2,2a)k+O(\log k)}, \qquad \binom{k/2}{k-bk-2ak} = 2^{\mathrm{h}(1/2,1-b-2a)k+O(\log k)},$$

$$\binom{k}{bk} = 2^{\mathrm{h}(b)k+O(\log k)}, \qquad \frac{\binom{2ak}{ak}}{2^{2ak}} = 2^{O(\log k)}.$$

Hence, with $\kappa = \mathrm{h}(b) - \mathrm{h}(1/2,2a) - \mathrm{h}(1/2,1-b-2a)$ as in the theorem,

$$p_0 = 2^{-\kappa k+O(\log k)}. \tag{3.5}$$

Since $\kappa > 4a$, this gives $p_0 \le 2^{-4ak}$ for all sufficiently large k; this comparison will be used for the left degrees.

Take $\lceil k^2/p_0 \rceil$ independent samples. A union bound over the at most 3^k queries gives probability $\le 3^k e^{-k^2}$ that some query is not covered. For any samples covering all queries, remove repeated pairs to obtain $\mathcal{B}$, and assign each query $\mathbf{y}$ exactly one pair $(B(\mathbf{y}), \boldsymbol{\varphi}(\mathbf{y})) \in \mathcal{B}$ with $\mathbf{y} \in \mathcal{Y}_{B(\mathbf{y}),\boldsymbol{\varphi}(\mathbf{y})}$. Below we will choose samples for which the left-degree upper bounds also hold.

The circuit. For each $(B, \boldsymbol{\varphi}) \in \mathcal{B}$, keep in $C_{B,\boldsymbol{\varphi}}$ only the output gates of the queries assigned to $(B, \boldsymbol{\varphi})$. Let C be the sum of these circuits: it shares the input and output gates and combines their disjoint middle gates, whose names are $(B, \boldsymbol{\varphi}, Q, \boldsymbol{\sigma})$; the coloring is inherited. Every query $\mathbf{y}$ receives edges only from $C_{B(\mathbf{y}),\boldsymbol{\varphi}(\mathbf{y})}$, so $(\mathbf{L}_C \mathbf{R}_C)_{\mathbf{x},\mathbf{y}} = f_{\mathbf{y}}(\mathbf{x})$ over the integers, and C computes $P_{k,1/2}$. Delete middle gates with no incoming or no outgoing edge. For every color (μ, θ), the right degree of each query is bounded as in Lemma 3.7:

$$\deg_{\mathsf{R}}(C_{(\mu,\theta)}) \le 2^{(b-1/2+2a+\mathrm{h}(a,\mu-a)+\mathrm{h}(\mu,\theta-\mu))k}.$$

The left degree. Fix a binary string $\mathbf{x}$ and a color (μ, θ). Before removing repeated pairs and restricting the output gates, the left degree of $\mathbf{x}$ in $C_{(\mu,\theta)}$ is at most $\sum_j D_j$, where for the j-th sample $(B_j, \boldsymbol{\varphi}_j)$ we put $D_j \coloneqq \deg_{\mathsf{L}}((C_{B_j})_{(\mu,\theta)}, \mathbf{x} \oplus \boldsymbol{\varphi}_j)$. Let $N_\mu = O^*\big(2^{(\mathrm{h}(1-b,\mu)-2\mu)k}\big)$ and $q_{\mu,\theta} = O^*\big(2^{(\mathrm{h}(1-b,\mu)+\mathrm{h}(2\mu,\theta)-4\mu)k}\big)$ denote the first two bounds of Lemma 3.7; they do not depend on B. Conditional on B_j, the vector $\boldsymbol{\varphi}_j$ is distributed as $\mathbf{z}$ in the lemma. Hence the variables D_j/N_μ are independent, lie in $[0,1]$, and have expectation at most $q_{\mu,\theta}/N_\mu$.

Put $M \coloneqq \lceil k^2/p_0 \rceil q_{\mu,\theta}/N_\mu + k^2$, which is at least the expectation of $\sum_j D_j/N_\mu$. The multiplicative Chernoff bound gives probability $\le e^{-M/3} \le e^{-k^2/3}$ that this sum exceeds $2M$. After a union bound over all 2^k binary strings and $\le (k+1)^2$ colors, this failure probability still tends to 0, as does the probability of an uncovered query. For all sufficiently large k, the combined failure probability is < 1, so there are samples for which every query is covered and all the upper bounds

$$\deg_{\mathsf{L}}(C_{(\mu,\theta)}) \le 2N_\mu M = 2\left\lceil \frac{k^2}{p_0} \right\rceil q_{\mu,\theta} + 2k^2 N_\mu \le \frac{1}{p_0}\, O^*\big(2^{(\mathrm{h}(1-b,\mu)+\mathrm{h}(2\mu,\theta)-4\mu)k}\big)$$

hold simultaneously; for the last inequality, the second term $2k^2 N_\mu = O^*\big(2^{(\mathrm{h}(1-b,\mu)-2\mu)k}\big)$ is absorbed into the first: since $1/p_0 \ge 2^{4ak} \ge 2^{2\mu k}$ and $\mathrm{h}(2\mu,\theta) \ge 0$, we have $2^{(\mathrm{h}(1-b,\mu)-2\mu)k} \le \frac{1}{p_0} 2^{(\mathrm{h}(1-b,\mu)+\mathrm{h}(2\mu,\theta)-4\mu)k}$. Removing repeated pairs and middle gates only decreases these counts. Fix such a family $\mathcal{B}$ and the query assignments.

Degree bounds. By (3.5), $1/p_0 = 2^{\kappa k + O(\log k)}$. Substituting this into the left-degree bound and comparing with the definitions of V_{L} and V_{R} gives

$$\begin{aligned}
\deg_{\mathsf{L}}(C_{(\mu,\theta)}) &\le 2^{(\kappa+\mathrm{h}(1-b,\mu)+\mathrm{h}(2\mu,\theta)-4\mu)k+O(\log k)} = 2^{V_{\mathsf{L}}(\mu,\theta)k+O(\log k)}, \\
\deg_{\mathsf{R}}(C_{(\mu,\theta)}) &\le 2^{V_{\mathsf{R}}(\mu,\theta)k},
\end{aligned}$$

with $V_{\mathsf{L}}, V_{\mathsf{R}}$ evaluated at the rounded parameters $a_k \coloneqq \lfloor ak \rfloor/k$ and $b_k \coloneqq \lfloor bk \rfloor/k$ in place of a and b. The colors (μ, θ) satisfy $\lfloor ak \rfloor \le \mu k \le 2\lfloor ak \rfloor$ and $\mu k \le \theta k \le 2\mu k$ with $\mu k, \theta k$ integers, so there are at most $(k+1)^2$ of them. To place them in the domain of the theorem, rename each color (μ, θ) to $(\mu, \theta) \cdot a/a_k$, a rescaling by $1 + O(1/k)$ that preserves $\mu \le \theta \le 2\mu$. Uniform continuity of the generalized entropy absorbs the floors and this rescaling into an $o(1)$ error in the exponents, uniform over colors. Hence $C_k \coloneqq C$ is the required circuit. □

Summing the degree bounds of Theorem 3.8 over the colors bounds the colored volume of C_k. Feeding this into the rebalancing Lemma 3.6, we obtain points of $\mathcal{K}(1/2)$, which is the form used in Section 4.

Corollary 3.9. *Let a, b and $V_{\mathsf{L}}, V_{\mathsf{R}}$ be as in Theorem 3.8, and define*

$$V(\tau) \coloneqq \max_{\substack{a \le \mu \le 2a \\ \mu \le \theta \le 2\mu}} \{\tau V_{\mathsf{L}}(\mu,\theta) + (1-\tau) V_{\mathsf{R}}(\mu,\theta)\} \qquad (\tau \in [0,1]).$$

Then V is nonnegative on $[0,1]$, and every pair $(l,r) \in \mathbb{R}^2$ with

$$\tau l + (1-\tau) r \ge \min\left\{V(\tau), \tau, \frac{1-\tau}{2}\right\} \qquad \text{for all } \tau \in [0,1]$$

belongs to $\mathcal{K}(1/2)$.

Proof of Corollary 3.9. Let C_k be the circuits of Theorem 3.8. Summing over its at most $(k+1)^2$ colors,

$$\begin{aligned}\rho_{C_k}(\tau) &= \sum_{(\mu,\theta)} \deg_{\mathsf{L}}\big((C_k)_{(\mu,\theta)}\big)^{\tau} \deg_{\mathsf{R}}\big((C_k)_{(\mu,\theta)}\big)^{1-\tau} \\ &\le (k+1)^2 \max_{(\mu,\theta)} 2^{(\tau V_{\mathsf{L}}(\mu,\theta) + (1-\tau) V_{\mathsf{R}}(\mu,\theta) + o(1))k} \le 2^{(V(\tau)+o(1))k}\end{aligned}$$

uniformly in $\tau \in [0,1]$. Since $P_{k,1/2}$ is nonzero, some middle gate of C_k has both an incoming and an outgoing edge, so $\rho_{C_k}(\tau) \ge 1$; letting $k \to \infty$ gives $V(\tau) \ge 0$. For the finitely many remaining even k, any colored circuit computing $P_{k,1/2}$ will do, since finitely many values of k do not affect the $o(1)$ term. The hypothesis of Lemma 3.6 is therefore satisfied, and that lemma gives the claim. □

3.5 A lower bound for OR circuits

We close the section by showing that the improvement of Section 3.4 cannot be achieved without algebraic cancellation. Recall that for $P_{n,1/2}$ the trivial circuit $(I, P_{n,1/2})$, in which each query enumerates all $2^{n/2}$ completions of its wildcards, is a PAR circuit with $(l,r) = (0,1/2)$, so $l+r = 1/2$. The following theorem shows that this is optimal even among OR circuits, whereas Corollary 3.9 gives $\mathsf{EXACT}_{\mathbb{Z}}$ circuits with $l + r < 1/2$.

Theorem 3.10. *Let $l, r \ge 0$, and suppose that there are* OR *circuits C_n computing $P_{n,1/2}$ with $\deg_{\mathsf{L}}(C_n) \le 2^{(l+o(1))n}$ and $\deg_{\mathsf{R}}(C_n) \le 2^{(r+o(1))n}$. Then $l + r \ge 1/2$.*

We remark that Burkhard, Fredman, and Kleitman [BFK81] and Fredman and Volper [FV82] proved this lower bound for PAR circuits. Their argument adapts directly to OR circuits; we include the proof for completeness.

Proof of Theorem 3.10. Let C be any OR circuit computing $P_{n,1/2}$, and fix $1/2 \le b \le 1$. Each middle gate of C defines the all-one rectangle whose rows and columns correspond to its adjacent input and output gates. Split the middle gates into two classes according to whether these rectangles contain at least $2^{n-\lfloor bn \rfloor}$ rows or fewer than this threshold.

- In the first class, at most $\lfloor bn \rfloor$ coordinates are constant across all binary strings $\mathbf{x}$ corresponding to the rows of a rectangle, since c constant coordinates leave at most 2^{n-c} strings. Every query in that rectangle must specify only such coordinates, with the constant values, so the rectangle has at most $\binom{\lfloor bn \rfloor}{n/2}$ columns. Each row belongs to at most $\deg_{\mathsf{L}}(C)$ rectangles of this class.

- In the second class, each rectangle has fewer than $2^{n-\lfloor bn \rfloor}$ rows, and each column belongs to at most $\deg_{\mathsf{R}}(C)$ such rectangles.

The matrix has 2^n rows, $\binom{n}{n/2}2^{n/2}$ columns, and $\binom{n}{n/2}2^n$ ones. Bound the first class's total area by summing over rows, and the second class's by summing over columns. Their rectangles cover all the ones, so dividing by the number of ones gives

$$\frac{\deg_{\mathsf{L}}(C)}{\binom{n}{n/2}/\binom{\lfloor bn \rfloor}{n/2}} + \frac{\deg_{\mathsf{R}}(C)}{2^{\lfloor bn \rfloor - n/2}} \geq 1.$$

If $r \geq 1/2$, then $l + r \geq 1/2$ trivially. Otherwise, take $b > 1/2 + r$, so that the second term tends to 0. Since $\binom{n}{n/2} = 2^{n+O(\log n)}$ and $\binom{\lfloor bn \rfloor}{n/2} \leq 2^{\mathrm{h}(b,1/2)n}$, the first term forces $l \geq 1 - \mathrm{h}(b, 1/2)$, and letting $b \downarrow 1/2 + r$ gives

$$l + r \geq g(r) \coloneqq 1 + r - \mathrm{h}(1/2 + r, 1/2).$$

On $[0, 1/2]$, we have $g'(r) = 1 - \log_2 \frac{1/2+r}{r} \leq 0$, so $g(r) \geq g(1/2) = 1 - \mathrm{h}(1, 1/2) + 1/2 = 1/2$. □

4 Putting everything together

Theorem 4.1. *Subset sum can be solved by a randomized algorithm in time* $O(2^{0.4999997n})$.

The proof chains together three results from the previous sections:

- Corollary 3.9 turns the function V into pairs $(l, r) \in \mathcal{K}(1/2)$;
- Corollary 3.3 shows that every such pair is an admissible exponent pair;
- Theorem 2.2 turns an admissible pair with $l + r = 1/2 - 2\xi$ into a subset sum algorithm in time $O(2^{(1/2-\xi/9+\varepsilon)n})$.

It remains to choose a, b, compute V, pick (l, r), and check numerically that the resulting ξ is large enough. Every quantity below has an explicit closed form, and its numerical value is listed in Table 1.

Parameters. We take

$$a \coloneqq 0.00037954865735953624129927, \qquad b \coloneqq 0.99094548027616196930017929,$$

obtained by numerical optimization. Clearly $a > 0$, $4a \leq 1 - b \leq 1/2$, and $1 - b > 10a$; the table shows $\kappa > 4a$. So Theorem 3.8 and Corollary 3.9 apply.

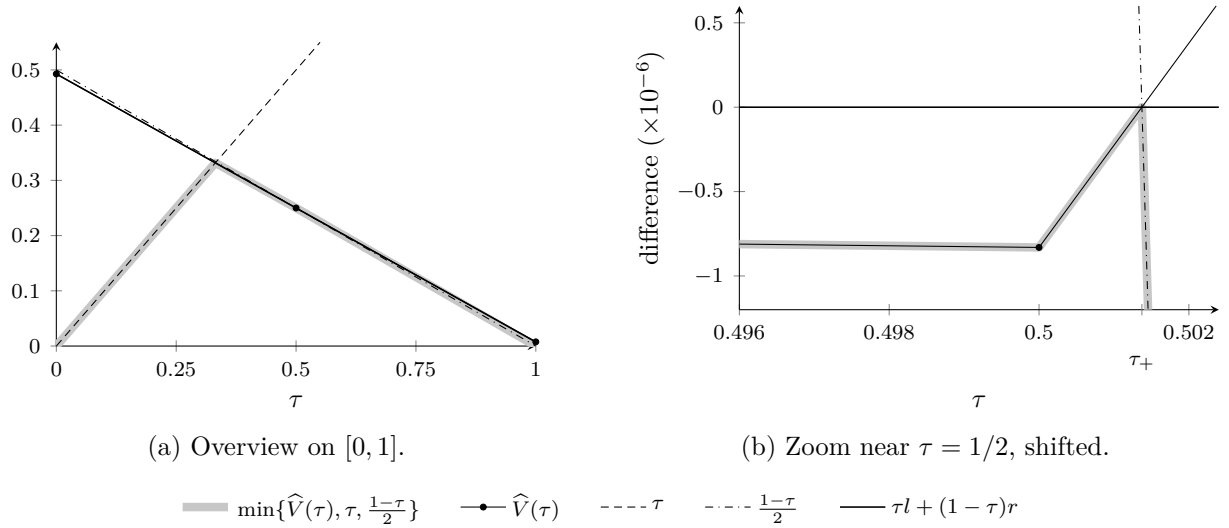
Three values of V. As a maximum of affine functions of τ, the function V is convex. We therefore only compute $V(0)$, $V(1/2)$, and $V(1)$, and bound V on $[0, 1]$ by the piecewise linear function $\widehat{V}$ through these three values.

- At $\tau = 0$, we maximize $\mathrm{h}(a, \mu - a) + \mathrm{h}(\mu, \theta - \mu)$. The maximum over θ is μ, at $\theta = 3\mu/2$, and then the maximum of $\mathrm{h}(a, \mu - a) + \mu$ is $a \log_2 6$, at $\mu = 5a/3$. Hence

$$V(0) = b - \frac{1}{2} + (2 + \log_2 6)a.$$

- At $\tau = 1$, we maximize $\mathrm{h}(1-b,\mu) + \mathrm{h}(2\mu,\theta) - 4\mu$. The maximum over θ is at $\theta = \mu$, leaving $\mathrm{h}(1-b,\mu) - 2\mu$. This is increasing for $\mu \le (1-b)/5$, hence on $[a, 2a]$ because $1 - b > 10a$. Taking $\mu = 2a$ gives
$$V(1) = \kappa + \mathrm{h}(1-b, 2a) - 4a.$$
- At $\tau = 1/2$, the maximum over θ of $\mathrm{h}(2\mu,\theta) + \mathrm{h}(\mu,\theta-\mu)$ is at $\theta^* := 4\mu/3$, where this sum minus 4μ equals $\mu \log_2(27/64)$. The remaining objective $\mathrm{h}(1-b,\mu)+\mathrm{h}(a,\mu-a)+\mu\log_2(27/64)$ is strictly concave in μ, and setting its derivative to zero gives
$$27(1-b-\mu)(2a-\mu) = 64\mu(\mu-a), \quad \text{i.e.} \quad 37\mu^2 + (27(1-b)-10a)\mu - 54a(1-b) = 0.$$
Its root $\mu^* \in (a, 2a)$ is the maximizer, and $V(1/2) = \frac{1}{2}(V_{\mathsf{L}}(\mu^*,\theta^*) + V_{\mathsf{R}}(\mu^*,\theta^*))$.

Choosing (l, r). By Corollary 3.9, we need a line $\tau l + (1-\tau)r$ lying above $\min\{\widehat{V}(\tau), \tau, (1-\tau)/2\}$ on $[0,1]$. Let τ_- be the point where the left piece of $\widehat{V}$ meets τ, and τ_+ the point where the right piece meets $(1-\tau)/2$:
$$\tau_- := \frac{V(0)}{1 + 2V(0) - 2V(1/2)}, \qquad \tau_+ := 1 - \frac{V(1)}{1/2 - 2V(1/2) + 2V(1)}.$$
Take the line through (τ_-, τ_-) and $(\tau_+, (1-\tau_+)/2)$, that is,
$$l := \frac{(1-\tau_+)(1-3\tau_-)}{2(\tau_+ - \tau_-)}, \qquad r := \frac{\tau_-(3\tau_+ - 1)}{2(\tau_+ - \tau_-)}.$$
On $[\tau_-, \tau_+]$ it is a chord of the convex function $\widehat{V}$, so it lies above $\widehat{V}$. Its slope $l - r$ lies in $(-1/2, 1)$, so it lies above τ to the left of τ_- and above $(1-\tau)/2$ to the right of τ_+. Hence $(l, r) \in \mathcal{K}(1/2)$, and by Corollary 3.3, (l, r) is an admissible exponent pair. Figure 1 illustrates these functions.



(a) Overview on $[0, 1]$.

(b) Zoom near $\tau = 1/2$, shifted.

Figure 1: The choice of (l, r). In (b), every curve is shifted by $-(\tau l + (1-\tau)r)$, so the requirement is that the envelope stays below 0.

Conclusion. Put $\xi := 1/4 - (l+r)/2$, so that $l + r = 1/2 - 2\xi$, and apply Theorem 2.2 with $\varepsilon := 10^{-9}$. By the table,

$$\frac{1}{2} - \frac{\xi}{9} + 10^{-9} = 0.49999777486376221650\ldots < \frac{1}{2} - \frac{1}{450000} = 0.499997\overline{7},$$

which proves Theorem 4.1.

Quantity	Value
κ	$0.00533321875848117877\ldots$
$4a$	$0.00151819462943814496\ldots$
$V(0)$	$0.49268569663735450584\ldots$
$V(1/2)$	$0.24997913372377961141\ldots$
$V(1)$	$0.00757766963945240565\ldots$
μ^*	$0.00069649211514613618\ldots$
θ^*	$0.00092865615352818158\ldots$
τ_-	$0.33168260605143320534\ldots$
τ_+	$0.50137304583751125366\ldots$
l	$0.00727587055962990941\ldots$
r	$0.49268405898808998771\ldots$
$(l+r)/2$	$0.24997996477385994856\ldots$
ξ	$0.00002003522614005143\ldots$
$1/2 - \xi/9$	$0.49999777386376221650\ldots$

Table 1: Numerical values at the chosen parameters.

AI disclosure and timestamps

In July and August 2026, we were developing a general theory of depth-2 linear circuits, building on [JS13, AL25, Ye26]. In August, while looking for applications of this theory, we found reductions from generalized subset sum (GSS) to compatibility testing for tensor powers of matrices, through the representation technique of [NW21, RW26]. In particular, we knew that GSS with $C = [\pm 2] := \{-2, -1, 1, 2\}$ could beat meet-in-the-middle if some circuit for the partial match matrix $P_{n,1/2}$ achieved $l + r < 1/2$.

On 5 September, we used GPT-6 Pro to construct the first circuit in our project meeting this criterion, a $\mathsf{SUP}_{\mathbb{Z}}$ circuit. From 5 to 28 September, we worked to incorporate this new GSS result into the existing theory. More generally, the reduction works for any fixed $s \in (0, 1/2]$ if some circuit for $P_{n,s}$ achieves $l + r < \mathrm{h}(s) - s$, where $\mathrm{h}(s) - s$ is the corresponding OR benchmark; this allowed us to shift from $s = 1/2$ to $s = 1/8$.

On 28 September, we used GPT-6 Astra to find a reduction showing that ordinary subset sum can also beat meet-in-the-middle using the same circuit for $P_{n,s}$. This gave our improved algorithm for subset sum; Record 1 arrives at the candidate exponent $0.499664500303\ldots$.

We then pursued the simplest self-contained proof, at a substantial cost in the exponent. For this, we shifted back to $s = 1/2$ and used an $\mathsf{EXACT}_{\mathbb{Z}}$ circuit instead. The manuscript locked on **4 October at 02:49** gives a randomized algorithm in worst-case time $O(2^{(1/2-1/450000)n})$, and Record 6 reports a computational reproduction at **03:51** that morning.

The table below lists, in chronological order, six records of our conversations with ChatGPT during the development of this work. Their timestamps are taken from the original message metadata, in Beijing time (UTC+8). These records were made public on **7 October 2026**.

Record	Link	Original message range, 2026
1	Computation Trees Criterion	Sep 16, 14:34–Sep 28, 06:48
2	View recent files	Sep 28, 07:16–07:31
3	View project conversations	Sep 28, 18:03–Sep 30, 23:39
4	Recover the improved P1 construction	Oct 1, 03:46–Oct 4, 02:49
5	Conceptual simplification	Oct 3, 05:57–19:07
6	Reproduce correctness check	Oct 4, 03:46–03:51

A preliminary version of this work was subsequently posted on arXiv [YL26] on 7 October 2026.

References

[AGP23] Josh Alman, Yunfeng Guan, and Ashwin Padaki. Smaller low-depth circuits for Kronecker powers. In *Proceedings of the 34th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4159–4187. Society for Industrial and Applied Mathematics, 2023. doi:10.1137/1.9781611977554.ch160. ↑7 , ↑18 , ↑21

[AKKN16] Per Austrin, Petteri Kaski, Mikko Koivisto, and Jesper Nederlof. Dense subset sum may be the hardest. In *Proceedings of the 33rd Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 47 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 13:1–13:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.STACS.2016.13. ↑3 , ↑6 , ↑11

[AL25] Josh Alman and Baitian Li. Kronecker powers, orthogonal vectors, and the asymptotic spectrum. In *Proceedings of the 66th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1411–1441. IEEE, 2025. doi:10.1109/FOCS63196.2025.00074. ↑7 , ↑8 , ↑15 , ↑18 , ↑20 , ↑21 , ↑30

[BBSS20] Xavier Bonnetain, Rémi Bricout, André Schrottenloher, and Yixin Shen. Improved classical and quantum algorithms for Subset-Sum. In Shiho Moriai and Huaxiong Wang, editors, *Proceedings of the 26th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, volume 12492 of *Lecture Notes in Computer Science*, pages 633–666. Springer, 2020. doi:10.1007/978-3-030-64834-3_22. ↑2

[BCJ11] Anja Becker, Jean-Sébastien Coron, and Antoine Joux. Improved generic algorithms for hard knapsacks. In Kenneth G. Paterson, editor, *Proceedings of the 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, volume 6632 of *Lecture Notes in Computer Science*, pages 364–385. Springer, 2011. doi:10.1007/978-3-642-20465-4_21. ↑2

[BFK81] Walter A. Burkhard, Michael L. Fredman, and Daniel J. Kleitman. Inherent complexity trade-offs for range query problems. *Theor. Comput. Sci.*, 16(3):279–290, 1981. doi:10.1016/0304-3975(81)90099-2. ↑27

[CJRS23] Xi Chen, Yaonan Jin, Tim Randolph, and Rocco A. Servedio. Subset Sum in time $2^{n/2}/\mathrm{poly}(n)$. In *Proceedings of Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, volume 275 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 39:1–39:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.APPROX/RANDOM.2023.39. ↑2

[Die96] Martin Dietzfelbinger. Universal hashing and k-wise independent random variables via integer arithmetic without primes. In Claude Puech and Rüdiger Reischuk, editors, *Proceedings of the 13th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 1046 of *Lecture Notes in Computer Science*, pages 567–580. Springer, 1996. doi:10.1007/3-540-60922-9_46. ↑10

[FV82] Michael L. Fredman and Dennis J. Volper. The complexity of partial match retrieval in a dynamic setting. *J. Algorithms*, 3(1):68–78, 1982. doi:10.1016/0196-6774(82)90009-8. ↑8 , ↑27

[HGJ10] Nick Howgrave-Graham and Antoine Joux. New generic algorithms for hard knapsacks. In Henri Gilbert, editor, *Proceedings of the 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, volume 6110 of *Lecture Notes in Computer Science*, pages 235–256. Springer, 2010. doi:10.1007/978-3-642-13190-5_12. ↑2 , ↑3 , ↑8 , ↑11

[HS74] Ellis Horowitz and Sartaj Sahni. Computing partitions with applications to the knapsack problem. *J. ACM*, 21(2):277–292, 1974. doi:10.1145/321812.321823. ↑2 , ↑11

[JS13] Stasys Jukna and Igor Sergeev. Complexity of linear boolean operators. *Found. Trends Theor. Comput. Sci.*, 9(1):1–123, 2013. doi:10.1561/0400000063. ↑7 , ↑15 , ↑18 , ↑21 , ↑30

[MNPW19] Marcin Mucha, Jesper Nederlof, Jakub Pawlewicz, and Karol Węgrzycki. Equal-Subset-Sum faster than the meet-in-the-middle. In *Proceedings of the 27th Annual European Symposium on Algorithms (ESA)*, volume 144 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 73:1–73:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ESA.2019.73. ↑2 , ↑3 , ↑4 , ↑8 , ↑11 , ↑13

[NW21] Jesper Nederlof and Karol Węgrzycki. 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)*, pages 1670–1683. ACM, 2021. doi:10.1145/3406325.3451024. ↑2 , ↑7 , ↑8 , ↑30

[Ope26] OpenAI. Subset Sum in time $O(2^{0.49n})$, 2026. Preprint dated 4 October 2026. URL: https://github.com/openai/math/blob/main/preprints/Subset-Sum-in-Time-2-power-0-49n-October-4-2026/subset-sum.pdf. ↑1 , ↑8

[RW26] Tim Randolph and Karol Węgrzycki. Beating meet-in-the-middle for subset balancing problems. In *Proceedings of the 58th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1314–1325. ACM, 2026. doi:10.1145/3798129.3800841. ↑2 , ↑3 , ↑4 , ↑8 , ↑11 , ↑30

[Ser22] Igor S. Sergeev. Notes on the complexity of coverings for Kronecker powers of symmetric matrices, 2022. arXiv:2212.01776. ↑7 , ↑18 , ↑21

[SS81] Richard Schroeppel and Adi Shamir. A $T = O(2^{n/2})$, $S = O(2^{n/4})$ algorithm for certain NP-complete problems. *SIAM J. Comput.*, 10(3):456–464, 1981. doi:10.1137/0210033. ↑2

[Str69] Volker Strassen. Gaussian elimination is not optimal. *Numer. Math.*, 13(4):354–356, 1969. doi:10.1007/BF02165411. ↑7

[Wil24] Ryan Williams. The orthogonal vectors conjecture and non-uniform circuit lower bounds. In *Proceedings of the 65th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1372–1387. IEEE, 2024. doi:10.1109/FOCS61266.2024.00088. ↑7

[Ye26] Lixi Ye. Improved depth-2 linear circuits for disjointness via quenched Lyapunov exponents, 2026. Preprint, version 2, 3 September 2026. arXiv:2603.15565v2. ↑30

[YL26] Lixi Ye and Baitian Li. Subset Sum via Partial Match, 2026. Preprint, version 1, 7 October 2026. arXiv:2610.10446v1. ↑31