Parameterized Algorithms on Integer Sets with Small Doubling: Integer Programming, Subset Sum and k-SUM

Tim Randolph
Harvey Mudd College
trandolph@hmc.edu

Karol Węgrzycki
Saarland University; Max Planck Institute for Informatics
wegrzycki@cs.uni-saarland.de

Abstract

We study the parameterized complexity of algorithmic problems whose input is an integer set A in terms of the *doubling constant* $\mathcal{C} := |A+A|/|A|$, a fundamental measure of additive structure. We present evidence that this new parameterization is algorithmically useful in the form of new results for two difficult, well-studied problems: Integer Programming and Subset Sum.

First, we show that determining the feasibility of bounded Integer Programs is a tractable problem when parameterized in the doubling constant. Specifically, we prove that the feasibility of an integer program $\mathcal{I}$ with n polynomially-bounded variables and m constraints can be determined in time $n^{O_{\mathcal{C}}(1)} \cdot \mathsf{poly}(|\mathcal{I}|)$ when the column set of the constraint matrix has doubling constant $\mathcal{C}$.

Second, we show that the Subset Sum and Unbounded Subset Sum problems can be solved in time $n^{O_{\mathcal{C}}(1)}$ and $n^{O_{\mathcal{C}}(\log \log \log n)}$, respectively, where the O_C notation hides functions that depend only on the doubling constant $\mathcal{C}$. We also show the equivalence of achieving an FPT algorithm for Subset Sum with bounded doubling and achieving a milestone result for the parameterized complexity of Box ILP. Finally, we design near-linear time algorithms for k-SUM as well as tight lower bounds for 4-SUM and nearly tight lower bounds for k-SUM, under the k-SUM conjecture.

Several of our results rely on a new proof that Freiman's Theorem, a central result in additive combinatorics, can be made efficiently constructive. This result may be of independent interest.

1 Introduction

Given a subset X of a group, the *doubling constant*

$$\mathcal{C} := \mathcal{C}(X) = \frac{|X + X|}{|X|}$$

is one measure used to capture the amount of "additive structure" in X. (Here, $X + X$ denotes the *sumset* $\{a + b \ : \ a, b \in X\}$.) This value ranges (on integer sets) from $2 - o_n(1)$ for arithmetic progressions to $\frac{n}{2} + o_n(1)$ when all sums are distinct, and is central to the study of additive combinatorics. If the doubling constant $\mathcal{C}$ is truly constant (that is, independent of the set cardinality $|X|$), this indicates that X is a "highly structured" set with respect to addition: for example, the statements that

- X has constant doubling ($|X + X| \leq c_1|X|$), that

- the iterated sumset $sX := \underbrace{X + X + X \cdots + X}_{s \text{ times}}$ is at most $c_2 := c_2(s)$ times $|X|$, and that

- X (likewise $X + X$ and sX) can be contained in a *generalized arithmetic progression* of dimension c_3 and volume $c_4|X|$,

are all equivalent up to the choice of constants c_1, c_2, c_3, and c_4 (c.f. [TV06] Proposition 2.26). The many fruitful applications of the doubling constant illustrate its significance as a robust measurement of additive structure (for an overview, see [TV06], especially Chapter 2).

In this work, we consider the parameterized complexity of problems on integer sets with respect to the doubling constant. Specifically, we focus on two problems for which additive structure is particularly helpful: Integer Programming and Subset Sum.

1.1 Related Work

Integer Programming and Subset Sum are not only problems in which additive structure plays an important role: they are also both well-studied and stubbornly difficult, to the point where significant work has gone into analysing their parameterized complexity and demarcating classes of tractable instances.

Integer Linear Programming. Many problems in *combinatorial optimization* can be formulated as an *integer linear program* (ILP). An ILP is an optimization problem of the following form:

$$\max \left\{ c^T x \mid Ax = b, x \in \mathbb{Z}_{\geq 0}^n \right\},$$

where $A \in \mathbb{Z}^{m \times n}$, $c \in \mathbb{Z}^n$ and $b \in \mathbb{Z}^m$. (ILPs of the form $Ax \geq b$ can be converted to this form using slack variables.) Unlike linear programming, integer programming is NP-complete. Due to its generality and both practical and theoretical importance, the complexity of ILP has been rigorously studied through the lens of *parameterized complexity*. Lenstra [Len83] has shown that an integer linear program with a fixed number of variables can be solved in polynomial time. His algorithm was subsequently improved, and the current record is $(\log n)^{O(n)}$ [RR23]. The question of whether this can be brought down to $2^{O(n)}$ is one of the most prominent open questions in the theory of algorithms.

ILP can also be parameterized in the number of constraints m and the maximum absolute value of any coefficient in the constraint matrix, $\Delta := \|A\|_\infty$. In 1981, Papadimitriou [Pap81] presented an $(m\Delta)^{O(m^2)}$-time algorithm, and the best algorithms for ILP parameterized in m and Δ continue to improve: see [JR18, EW19] for recent progress. Another class of tractable instances of ILP rely on structural properties of the constraint matrix (see [CKL$^+$24, KKM20, CEH$^+$21, CEP$^+$21]).

Subset Sum. Along with the closely related Knapsack problem, the Subset Sum problem is the canonical NP-complete problem concerning addition in integer sets. In addition to NP-completeness, the problem appears difficult from the standpoint of exact algorithms: despite significant attention (see, e.g., [Woe08, AKKN16, NW21]), solving Subset Sum in time $2^{(1/2-c)n}$ for some constant $c > 0$ remains a major open problem. Except for "log shaving" results that improve runtime by subexponential factors [CJRS23], the exact runtime has not been improved in 50 years [HS74]. The lack of progress in exact algorithms motivates parameterized approaches, including a long line of pseudopolynomial-time algorithms parameterized by the size of the target [Bri17, KX19, ABHS22] and the largest input integer [EW19, BW21, PRW21, CLMZ23].

However, these parameterized results do not take advantage of structural properties of the input when the input numbers are very large. Therefore, we complement the parameterization based on the input size by considering the parameterized complexity of Subset Sum with respect to the doubling constant. This choice is natural not only because the doubling constant is essential to the study of integer sets under addition, but also because existing results from additive combinatorics give strong bounds on the search space: Freiman's Theorem bounds the number of distinct subset sums of an n-element input set by $n^{f(\mathcal{C})}$, where f is a function that depends only on $\mathcal{C}$.

The parameterization of Subset Sum in the cardinality of the solution k, otherwise known as k-SUM, has an entire literature of its own. Simple "meet-in-the-middle" algorithms that run in time $O(n^{\lceil k/2 \rceil})$ are conjectured to be optimal up to polynomial factors. The results of Abboud, Bringmann, and Fischer, and of Jin and Xu, suggest that the hardest instances of k-SUM are those with very little additive structure, such as Sidon sets [ABF23, JX23]. Parameterizing k-SUM in the doubling constant allows us to make analogous conclusions for the more general case of k-SUM: we can now prove results of the form, "k-SUM instances with strong additive structure (i.e., small doubling constant) are easy".

Algorithms and Additive Combinatorics. This paper is also motivated by an emerging trend in fine-grained complexity and algorithms: "importing" results from additive combinatorics. In several recent works, researchers have achieved breakthroughs by taking existential results from the field of additive combinatorics and modifying their proofs to make them explicitly and efficiently constructive.

For example, in 2015 Chan and Lewenstein proved a version of the Balog-Szemeredi-Gowers (BSG) theorem that allows certain sets guaranteed by the theorem to be constructed algorithmically [CL15]. They then leveraged this result to solve the $(\min, +)$-convolution and 3-SUM problems on monotone sets of small integers. Recently, the constructive BSG theorem found new applications. In 2022, Abboud, Bringmann and Fischer used this result, as well as a constructive version of Ruzsa's covering lemma, as a key ingredient in their proofs of lower bounds for approximate distance oracles and listing 4-cycles [ABF23]. In the same year, Jin and Xu independently proved similar lower bounds and used the constructive BSG theorem to reduce 3-SUM to 3-SUM on Sidon sets [JX23]. More broadly, these works reflect the increasing role of additive combinatorics in algorithms over the last few decades; for general references, see [Tre09, Vio11, Bib13, Lov17].

1.2 Our Results

Contribution 1: *A Constructive Freiman's Theorem in Near-Linear FPT Time.* We begin by unlocking a new tool to help us manipulate sets with significant additive structure. *Freiman's Theorem*, a cornerstone result in additive combinatorics, states that every integer set with constant doubling is contained inside a small (generalized) arithmetic progression. Naively constructing this generalized arithmetic progression takes XP-time. We make the construction efficient by showing how an algorithm can obtain such an arithmetic progression in time $\widetilde{O}_{\mathcal{C}}(n)$[1] (Theorem 3.2). Later in the paper, we use this theorem to reduce Subset Sum with constant doubling to a constrained integer programming problem (Contribution 3) and to design efficient algorithms for Unbounded Subset Sum (Contribution

---

[1]We use $O_{\mathcal{C}}$ notation to indicate the suppression of terms that depend only on $\mathcal{C}$. For example, $O_{\mathcal{C}}(n^2) = f(\mathcal{C}) \cdot O(n^2)$ for some computable function f. $\widetilde{O}$ hides factors polylogarithmic in the argument, in this case $\log(n)$.

4). We hope that, like the constructive BSG theorem discussed above, the constructive statement of Freiman's Theorem may find other independent applications.

Contribution 2: *Integer Programming with Constant Doubling.* An integer program specified by a constraint matrix $A \in \mathbb{Z}^{m \times n}$ and solution vector $b \in \mathbb{Z}^m$ is *feasible* if there exists a solution $x \in \mathbb{Z}_{\geq 0}^n$ such that $Ax = b$. The ILP is *binary* if the variables are further restricted to $x \in \{0,1\}^n$.

In our setting, we consider integer programs in which the set of column vectors

$$\mathcal{A} \coloneqq \{A[\cdot, j] \mid j \in [n]\}$$

has constant doubling: $|\mathcal{A} + \mathcal{A}| \leq \mathcal{C}|\mathcal{A}|$, for a constant $\mathcal{C}$. We prove the following:

Theorem 1.1. *An instance $\mathcal{I}$ of $\mathcal{C}$-Binary ILP Feasibility on n variables can be solved in time $n^{O_\mathcal{C}(1)} \cdot \mathsf{poly}(|\mathcal{I}|).$*[2]

This follows from Freiman's Theorem (without construction) and a dynamic programming algorithm. The theorem also holds when the variables $x_1, x_2, \ldots, x_n$ have upper and lower bounds of magnitude $\mathsf{poly}(n)$.

Contribution 3: *Subset Sum with Constant Doubling.* Our result for integer programming with constant doubling implies an $n^{O_\mathcal{C}(1)}$-algorithm for Subset Sum (Corollary 5.1).

Assuming the Exponential Time Hypothesis (ETH), there is no $2^{o(n)}$ time algorithm for Subset Sum. Because $\mathcal{C} = O(n)$, this means that we cannot hope for a $2^{o(\mathcal{C})} n^{o(\mathcal{C}/\log(\mathcal{C}))}$ algorithm for $\mathcal{C}$-Subset Sum under the ETH. However, this lower bound does not exclude an $2^{O(\mathcal{C})} \cdot n^{O(1)}$ algorithm. A natural question is thus whether our upper bound can be improved to an Fixed-Parameter Tractable (FPT) result: can $\mathcal{C}$-Subset Sum be solved in time $O_\mathcal{C}(\mathsf{poly}(n))$? We show that this result appears unlikely by way of an interesting connection to the feasibility of integer programs with binary variables.

Theorem 1.2. *$\mathcal{C}$-Subset Sum can be solved in time $O_\mathcal{C}(\mathsf{poly}(n))$ if and only if Hyperplane-Constrained Binary ILP (HBILP) can be solved in time $\Delta^{O(m)} \cdot O_m(\mathsf{poly}(|\mathcal{I}|))$, where $|\mathcal{I}|$ is the size of the instance.*

HBILP considers a constraint matrix $A \in \mathbb{Z}^{m \times n}$ with entries bounded by $\Delta \coloneqq \|A\|_\infty$, and asks whether there exists a solution $x \in \{0,1\}^n$ such that $\langle Ax, s \rangle = t$ for a certain target t and "step vector" s orthogonal to a hyperplane. The best existing algorithm solves HBILP Feasibility in time $O_m(|\mathcal{I}|) + \Delta^{O(m^2)}$ ([DLRV23], Corollary 1[3]).

We can also reduce ILP Feasibility with bounded variables to HBILP feasibility (Lemma 5.4). Thus Theorem 1.2 implies that an FPT algorithm for Subset Sum with constant doubling would imply a $\Delta^{O(m)} \cdot \mathsf{poly}(n)$ algorithm for ILP Feasibility with bounded variables (Corollary 5.2). As previously noted in [DLRV23], reducing the exponent of Δ from $O(m^2)$ to $O(m)$ would be analogous to the recent improvement achieved by Eisenbrand and Weismantel for integer programs with *unbounded* variables [EW19].

Such an algorithm for ILP Feasibility would resolve the feasibility portion of one of the most significant open questions in the parameterized complexity of integer programming: whether the $(\Delta^{O(m)} \cdot O_m(|\mathcal{I}|))$-time algorithm for ILPs with unbounded variables can be extended to ILPs with bounded variables [EW19, JR18, KPW20]. This would be a significant breakthrough in the area [JR18, KPW20]; accordingly, finding an FPT algorithm for $\mathcal{C}$-Subset Sum is at least as difficult.

.

---

[2]We write $|\mathcal{I}|$ to denote the size of the ILP instance $\mathcal{I}$. In the word RAM model (see Section 2), this is $\mathsf{poly}(m, n)$.

[3]Corollary 2 in the arXiv preprint, 2303.02474.

Contribution 4: *Unbounded Subset Sum with Constant Doubling.* We can reduce an instance of Unbounded Subset Sum with constant doubling to an ILP with m constraints, n binary variables, and entries of A bounded by $\Delta = n^{O(1/d(\mathcal{C}))}$ using our constructive Freiman's theorem. Because solvable ILPs with bounded Δ admit solutions with small support, this allows us to solve Unbounded Subset Sum in time $n^{O_{\mathcal{C}}(\log \log \log n)}$, or $n^{O_{\mathcal{C}}(1)}$ under the hypothesis that a v-variable ILP $\mathcal{I}$ can be solved in time $2^{O(v)}\mathsf{poly}(\mathcal{I})$ (Theorem 6.1).

Contribution 5: *k-SUM with Constant Doubling.* The application of recent algorithms for sparse nonnegative convolution [BFN22] allow us to efficiently solve k-SUM with constant doubling in time $\widetilde{O}(\mathcal{C}^{\lceil k/2 \rceil} \cdot 2^{O(k)} \cdot n)$ (see Theorem 7.1).

Because the k-SUM conjecture implies a lower bound of $\Omega(\mathcal{C}^{\lceil k/2 \rceil - 1}n)$, this leaves a $\mathcal{C}$-factor gap. Part of the gap can be explained by the fact that the Plünnecke-Ruzsa inequality, which we use to derive the upper bound, does not give the optimal exponent for $\mathcal{C}$; applying recent improvements to the inequality narrows the gap slightly. In the specific case of $(\mathcal{C}, 4)$-SUM, our algorithm achieves a runtime of $\widetilde{O}(\mathcal{C}n)$, which is optimal up to polylogarithmic factors under the k-SUM conjecture.

1.3 Organization

We begin with mathematical preliminaries in Section 2, although some definitions required for the constructive proof of Freiman's Theorem in Section 3 are deferred to the proof of this result in Appendix A. In Section 4, we present our algorithms for ILP feasibility with bounded doubling. Finally, we present our bounds for Subset Sum in Section 5, Unbounded Subset Sum in Section 6, and k-SUM in Section 7.

2 Preliminaries

RAM Model. Throughout the paper, we use the standard *word RAM* model, in which input integers fit into a single machine word and logical and arithmetic operations on machine words take time $O(1)$. If we make the weaker assumption that operations on b-bit words take $\mathsf{polylog}(b)$ time, this adds a $\mathsf{polylog}(b)$ factor to Theorem 3.2 and the results that rely on it.

Big-O Notation. We use $O_{\mathcal{C}}$ notation to indicate we have suppressed terms that depend only on $\mathcal{C}$. For example, $O_{\mathcal{C}}(n^2) = f(\mathcal{C}) \cdot O(n^2)$ for some computable function f. $\widetilde{O}$ notation suppresses polylogarithmic factors of n and Δ: for instance, $n\log^2(n) = \widetilde{O}(n)$.

Sets. We write $[n]$ for the integer set $\{1, 2, \ldots, n\}$ and $[a : b]$ (with $a \leq b$) for the integer set $[a, a + 1, a + 2, \ldots, b]$. The *diameter* of an integer set A, denoted $\mathrm{diam}(A)$, is $\max_{a,b \in A} |a - b|$. We write $\Sigma(X)$ as shorthand for the sum of elements $\sum_{x \in X} x$, and $\Sigma(2^X)$ as shorthand for the set of subset sums $\{\Sigma(X') \ : \ X' \subseteq X\}$.

Vectors. Given a vector $x \in \mathbb{Z}^n$, we define $\mathrm{supp}(x) \subseteq [n]$ to be the set of non-zero coordinates of x.

For $a, b \in \mathbb{Z}_{\geq 0}^n$ we say that a is *lexicographically prior* to b, denoted $a \prec_{\mathrm{lex}} b$, if and only if there exists $k \in [n]$ such that $a[k] < b[k]$ and for every $1 \leq i < k$ it holds that $a_i = b_i$. Observe that $\prec_{\mathrm{lex}}$ is a total order and that every set of vectors $S \subseteq \mathbb{Z}_{\geq 0}^n$ contains a unique element that is lexicographically minimal.

Matrices. Given a $m \times n$ matrix A, we write $A[i, j]$ to denote the component of A at row i, column j. We write $A[i, \cdot]$ and $A[\cdot, j]$ to denote the ith row and jth column of A, respectively.

We write $J_{m \times n}$ to denote the $m \times n$ matrix in which each entry is 1.

Group Theory and Linear Algebra. Given an integer m, we write $\mathbb{Z}_m$ to denote the cyclic group of order m (under addition). When p is prime, every element of $\mathbb{Z}_p$ is a generator except for 0.

A *lattice* in $\mathbb{R}^d$ is defined by d linearly independent vectors $v_1, v_2, \ldots, v_d \in \mathbb{R}^d$, collectively referred to as the *basis* of the lattice. The lattice itself is the set

$$\Lambda = \left\{ \sum_{i \in [d]} a_i v_i \;\middle|\; a_i \in \mathbb{Z} \right\}$$

of all integer linear combinations of $v_1, v_2, \ldots, v_d$, and each point in Λ is referred to as a *lattice vector*.

The *determinant* of a lattice, denoted $\det(\Lambda)$, is the determinant of the matrix whose columns are the lattice basis. Geometrically, $\det(\Lambda)$ is the volume of the *fundamental parallelepiped* spanned by the lattice basis. In general, if T is a convex body, we write $vol(T)$ to denote the volume of T.

Given two m-dimensional vectors x and y, we use $\langle x, y \rangle$ to denote the dot product $x_1 y_1 + \cdots + x_m y_m$.

Norms. Given a real number r, we write $\|r\|_{\mathbb{R}/\mathbb{Z}}$ to denote the distance from the nearest integer. Given a finite-dimensional vector v, the L_∞ norm $\|v\|_\infty$ denotes the largest absolute value of any coordinate.

Additive Combinatorics. Given an integer set X, $X + X$ denotes the *sumset* $\{a + b \; : \; a, b \in X\}$. We write sX, where s is a positive integer, as shorthand for the iterated sumset $\underbrace{X + \ldots + X}_{s \text{ times}}$.

A *generalized arithmetic progression* (GAP) P is an integer set

$$P = \{\ell_1 y_1 + \ell_2 y_2 + \cdots + \ell_d y_d \; : \; 0 \le \ell_i < L_i, \forall i \in [d]\},$$

defined by the integer vector $y = \{y_1, y_2, \ldots, y_d\}$ and the dimension bounds $L_1, L_2, \ldots, L_d$. We say that P has *dimension d* and *volume* $\prod_{i \in [d]} L_i$. When we write that an algorithm "explicitly constructs" or "returns" P, we mean specifically that the algorithm computes y_i and L_i for all $i \in [d]$.

We can think of P as a projection of a d-dimensional parallelepiped onto the line. P is *proper* if $|P| = L_1 L_2 \ldots L_d$, that is, if each point in the parallelepiped projects to a unique point on the line.

3 Freiman's Theorem Made Constructive in FPT Time

Freiman's Theorem states that any integer set X with constant doubling is contained inside a generalized arithmetic progression of constant dimension and volume at most $|X|$ times a constant.

Theorem 3.1 (Freiman's Theorem, [Fre64], see [Zha22] for a modern presentation)**.** *Any finite integer set X with $|X + X| \le \mathcal{C}|X|$ is contained in a GAP P of dimension $d(\mathcal{C})$ and volume $v(\mathcal{C})|X|$, where d and v are computable functions that depend only on $\mathcal{C}$.*

We make this statement constructive by showing an algorithm that, given X, can explicitly construct the progression P in FPT time. In fact, the construction is near-linear, losing only a $\mathsf{polylog}(n)$ factor and a (large) function of $\mathcal{C}$.

Theorem 3.2 (FPT Freiman's Theorem)**.** *Let A be a set of n integers satisfying $|A + A| \le \mathcal{C}|A|$. There exists an $\widetilde{O}_\mathcal{C}(n)$ algorithm that, with probability $1 - n^{-\gamma}$ for an arbitrarily large constant $\gamma > 0$, returns[4] an arithmetic progression*

$$P = \{x_1 \ell_1 + x_2 \ell_2 + \cdots + x_{d(\mathcal{C})} \ell_{d(\mathcal{C})} \; : \; \forall i, \ell_i \in [L_i]\} \supseteq A$$

with dimension $d(\mathcal{C})$ and volume $v(\mathcal{C}) \cdot |A|$, where d and v are computable functions that depend only on $\mathcal{C}$.

(We make the standard assumption that arithmetic operations on integers require $O(1)$ time.)

In outline, the proof proceeds as follows:

---

[4]Specifically, we compute the values $x_1, x_2, \ldots, x_{d(\mathcal{C})}$ and $L_1, L_2, \ldots L_{d(\mathcal{C})}$.

- *Step 1*: We prove a constructive version of *Ruzsa's Modeling Lemma*, which allows us to map our integer set to a small cyclic group such that additive structure is preserved.
- *Step 2*: We prove a constructive version of *Bogolyubov's Lemma*, which allows us to find a Bohr set contained in the cyclic group. Roughly speaking, the Bohr set (1) behaves "like a subspace" and (2) is within a constant factor of the size of our original set. This step requires the Fast Fourier Transform (FFT).
- *Step 3*: Finding a large GAP within our Bohr set requires finding a small basis for a certain lattice. Fortunately, the lattice has dimension $O_{\mathcal{C}}(1)$, so we can enumerate the entire set of short lattice vectors.
- *Step 4*: At this point we are left with a GAP that covers a constant fraction of the image of our original set in the cyclic group. Using *Ruzsa's Covering Lemma*, previously made efficiently constructive by Abboud, Bringmann, and Fischer [ABF23], we can find a GAP that covers the entire image of our input set. We then map back to the integers to complete the construction.

We defer the full proof to Appendix A. The following observation further simplifies Theorem 3.2.

Observation 3.1. In the GAP P guaranteed by Theorem 3.2, without loss of generality we can assume

$$L_i \leq n^{2/d(\mathcal{C})}$$

for all $i \in [d(\mathcal{C})]$, where $d(\mathcal{C})$ denotes the dimension of P.

We defer the proof of Observation 3.1 to Appendix A.6.

4 Integer Programming with Constant Doubling

For an integer program, we consider the doubling constant of the *column set* of the constraint matrix A as our parameter. This is because the column is the smallest unit affected by each variable x_i when we compute the product Ax; as a result, duplicate columns in A play a similar role to duplicate elements in a Subset Sum instance, and indeed can often be eliminated without loss of generality. This formulation allows A to contain duplicate entries (for example, multiple 0's and 1's) as long as all columns are distinct.

Given a matrix A, we use the shorthand

$$\mathcal{A} := \mathcal{A}(A) = \{A[\cdot, j] \mid j \in [n]\}$$

to denote the set of column vectors of A. Vector set addition (that is, $\mathcal{A} + \mathcal{A}$) is defined in the natural way, using vector instead of integer addition.

Problem 1: $\mathcal{C}$-Integer Linear Programming (ILP) Feasibility

In: An integer linear program specified by an integer matrix $A \in \mathbb{Z}^{m \times n}$ with n distinct columns and an integer target $b \in \mathbb{Z}^m$, such that the column set $\mathcal{A} := \mathcal{A}(A)$ satisfies $|\mathcal{A} + \mathcal{A}| \leq \mathcal{C}|\mathcal{A}|$ for a constant $\mathcal{C}$ independent of m and n.
Out: Vector $x \in \mathbb{Z}_{\geq 0}^n$ such that $Ax = b$, or 'NO' if no solution exists.

If each variable x_i is constrained to satisfy $x_i \in [\ell_i : u_i]$, where ℓ_i and u_i indicate the lower and upper bounds of a range of valid variable assignments, we refer to the problem as $\mathcal{C}$-**Bounded ILP Feasibility**. Further restricting the variables to $x \in \{0, 1\}^n$ yields $\mathcal{C}$-**Binary ILP Feasibility**.

Remark 4.1. Bounded ILPs with n variables and $|\ell_i|, |u_i| = O(\mathsf{poly}(n))$ for $i \in [n]$ can be converted into equivalent binary ILPs with $\mathsf{poly}(n)$ variables by duplicating the columns of A.

4.1 $\mathcal{C}$-Binary ILP Feasibility

Given a constraint matrix with constant doubling, Freiman's Theorem bounds the number of possible values for Ax corresponding to any variable assignment if the variables are binary or bounded. This allows us to solve the problem efficiently via dynamic programming, and does not actually require constructing the GAP guaranteed by Freiman's Theorem.[5]

Theorem 1.1. *An instance $\mathcal{I}$ of $\mathcal{C}$-Binary ILP Feasibility on n variables can be solved in time $n^{O_\mathcal{C}(1)} \cdot$* $\mathsf{poly}(|\mathcal{I}|)$.[6]

Proof. Fix an instance of $\mathcal{C}$-Binary ILP feasibility specified by $A \in \mathbb{Z}^{m \times n}$ and $b \in \mathbb{Z}^m$, with the column set $\mathcal{A}$ satisfying $|\mathcal{A} + \mathcal{A}| \leq \mathcal{C}|\mathcal{A}|$.

Let $L := \Sigma(2^\mathcal{A})$ denote the list of all (vector) sums that can be attained by adding together any subset of the columns of A. Equivalently, this is the set of possible outputs Ax for any $x \in \{0,1\}^n$. Our first goal is to bound $|L|$.

First, we observe that there exists a GAP P with dimension $d(\mathcal{C})$ and volume $v(\mathcal{C})n$ such that

$$\mathcal{A} \subseteq P = \{x_1 k_1 + x_2 k_2 + \cdots + x_{d(\mathcal{C})} k_{d(\mathcal{C})} \ : \ \forall i, k_i \in [K_i]\},$$

where $x_i \in \mathbb{Z}^m$ for all $i \in [d(\mathcal{C})]$. This is true even though $\mathcal{A}$ is a set of integer vectors, as Freiman's Theorem holds for torsion-free[7] commutative groups ([Ruz09], Theorem 8.1).

Thus L is contained in the GAP

$$P' = \{x_1 k_1 + x_2 k_2 + \cdots + x_{d(\mathcal{C})} k_{d(\mathcal{C})} \ : \ \forall i, k_i \in [n \cdot K_i]\},$$

which implies

$$|L| \leq |P'| \leq n^{d(\mathcal{C})} |P| = n^{d(\mathcal{C})} v(\mathcal{C}) n = n^{O_\mathcal{C}(1)}. \tag{1}$$

To complete the proof, we claim that we can enumerate L efficiently via dynamic programming, using the following procedure: Initially, we set $L_1 = A[1, \cdot]$. Then, we iterate $i = 2, 3, \ldots, n$. In the ith iteration, we construct the sorted list L_i, defined as:

$$L_i := L_{i-1} \cup \{a + A[i, \cdot] \mid a \in L_{i-1}\}.$$

Finally, we return list $L = L_n$. Correctness of the above algorithm follows immediately by a construction. For the running time, observe that L_i can be constructed in $O(|L_i|)$ time. Because each of the n iterations of the subprocedure takes time $O(|L_i|) = O(|L|)$, the total runtime is, by (1), at most $n \cdot O(|L|) = n^{O_\mathcal{C}(1)}$. $\quad\square$

4.2 $\mathcal{C}$-Bounded ILP Feasibility

In general, ILPs with polynomially bounded variables can be converted to ILPs with binary variables (see Remark 4.1); however, the straightforward reduction can create many duplicate columns in the resulting Binary ILP. Although it is possible to get rid of the duplicate columns, it is easier to extend the previous result to $\mathcal{C}$-Bounded ILP Feasibility directly:

Corollary 4.1. *An instance $\mathcal{I}$ of $\mathcal{C}$-Bounded ILP Feasibility such that $\ell_i \leq x_i \leq u_i$ and $|\ell_i|, |u_i| =$* $\mathsf{poly}(n)$ *for $i \in [n]$ can be solved in time $n^{O_\mathcal{C}(1)} \cdot \mathsf{poly}(|\mathcal{I}|)$.*

Proof. Modify the proof of Theorem 1.1 by considering the list L' of all possible outputs Ax for each valid assignment of variables x, using the variable bounds $x_i \in [\ell_i : u_i]$ for $i \in [n]$ instead of $x \in \{0,1\}^n$. As before, we bound $|L'|$.

---

[5]The constructive Freiman's theorem will be required later, specifically in Lemma 5.2 and Theorem 6.1. The current result emphasizes the usefulness of parameterization in the doubling constant.

[6]We write $|\mathcal{I}|$ to denote the size of the ILP instance $\mathcal{I}$. In the word RAM model (see Section 2), this is $\mathsf{poly}(m, n)$.

[7]That is, groups in which only the identity element has finite order.

Observe that L' is contained in the GAP P'' obtained by scaling each range bound L_i of P by a factor of $n^{O(1)}$, where the hidden constant is determined by the bounds on the variables. It follows that $|L'| = n^{O_{\mathcal{C}}(1)}$. We can enumerate L' by modifying the procedure given above so that Step 2 merges a polynomial number of lists, one for each variable assignment. $\qquad\square$

5 Subset Sum with Constant Doubling

We now consider the useful applications of parameterization in the doubling constant to Subset Sum. Formally, we consider the following problem:

Problem 2: $\mathcal{C}$-Subset Sum

In: An integer set $Z = \{z_1, z_2, \ldots, z_n\}$ such that $|Z + Z| \leq \mathcal{C}|Z|$ and an integer target t.
Out: $S \subseteq Z$ such that $\Sigma(S) = t$, or 'NO' if no solution exists.

$\mathcal{C}$-Subset Sum is equivalent to $\mathcal{C}$-Binary ILP with a single constraint. As a result, Theorem 1.1 yields the following corollary for Subset Sum with n variables:

Corollary 5.1 ($\mathcal{C}$-Subset Sum is in XP). *$\mathcal{C}$-Subset Sum can be solved in time $n^{O_{\mathcal{C}}(1)}$.*

At this point, it is natural to wonder whether $\mathcal{C}$-Subset Sum can be solved in time $O_{\mathcal{C}}(1) \cdot n^{O(1)}$: that is, whether Subset Sum is in FPT with respect to the doubling constant. While we cannot yet prove or disprove this statement, we can show that it is equivalent to an open problem in the parameterized complexity of integer programming. The remainder of this section proves this reduction in both directions.

5.1 Reduction from $\mathcal{C}$-Subset Sum to Hyperplane-Constrained Binary ILP Feasibility

Recent generalizations of Integer Programming consider the problem of optimizing the value $g(Ax)$ in place of Ax, where $g : \mathbb{R}^m \to \mathbb{R}$ is a low-dimensional objective function [DLRV23]. The mapping given by Freiman's Theorem provides a natural reduction from Subset Sum with constant doubling to a problem of this form. Specifically, $\mathcal{C}$-Subset Sum reduces to a Binary ILP feasibility problem in which the constraint matrix A has bounded entries and a feasible solution is any x satisfying $\langle Ax, s \rangle = t$ for a specific "step vector" s. Formally, our problem is as follows:

Problem 3: Hyperplane-Constrained Binary ILP (HBILP) Feasibility

In: An integer matrix $A \in \mathbb{Z}^{m \times n}$, a step vector $s \in \mathbb{Z}^m$, and a target integer t. We let $\Delta := \|A\|_\infty$, the magnitude of A's largest entry.
Out: A vector $x \in \{0, 1\}^n$ such that $\langle Ax, s \rangle = t$, or 'NO' if no solution exists.

The reduction from $\mathcal{C}$-Subset Sum to HBILP Feasibility (Lemma 5.2) is straightforward but relies crucially on our constructive Freiman's Theorem.

Lemma 5.1. *For any fixed instance (Z, t) of $\mathcal{C}$-Subset Sum, there exists a HBILP Feasibility instance given by $A \in \mathbb{Z}^{d(\mathcal{C}) \times n}$, $s \in \mathbb{Z}^{d(\mathcal{C})}$, and t for some function $d(\mathcal{C})$ such that a vector $x \in \{0, 1\}^n$ satisfies*

$$Ax = t \quad \text{if and only if} \quad \sum_{i\,:\,x_i=1} z_i = t.$$

Moreover, $\Delta := \|A\|_\infty \leq n^{2/d(\mathcal{C})}$, and the reduction can be computed in time $\widetilde{O}_{\mathcal{C}}(n)$ with success probability $1 - n^{-\gamma}$ for an arbitrarily small constant γ.

Proof. Fix an instance of $\mathcal{C}$-Subset Sum given by an integer set Z satisfying $|Z + Z| \leq \mathcal{C}|Z|$ and an integer target t. Apply Theorem 3.2, which fails with probability $n^{-\gamma}$ and otherwise produces a GAP

$$P = \{y_1\ell_1 + y_2\ell_2 + \cdots + y_d\ell_d \ : \ \forall i, \ell_i \in [L_i]\}$$

of dimension $d := d(\mathcal{C})$ and volume $v(\mathcal{C})n$ containing Z.

For each $z_i \in Z$, let $v(z_i) = (v_1, v_2, \ldots, v_d)$ be an arbitrary $d(\mathcal{C})$-dimensional integer vector satisfying

$$y_1v_1 + y_2v_2 + \ldots y_dv_d = z_i \text{ and}$$
$$\forall i \in [d], v_i \in [L_i].$$

We can think of $v(z_i)$ as the d-dimensional "GAP coordinates" of the input element z_i. $v(z_i)$ is guaranteed to exist by Freiman's theorem, and we can recover it in time $O(|P|) = O_{\mathcal{C}}(n)$ via exhaustive search of P. (However, $v(z_i)$ is not guaranteed to be unique.)

To complete the reduction, set

$$A \in \mathbb{Z}^{d(\mathcal{C}) \times n} \text{ with } \forall j \in [n], A[\cdot, j] = v(z_j),$$

set $s := (y_1, y_2, \ldots, y_d)$ and preserve the same target t. Note that $\|A\|_\infty \leq n^{2/d(\mathcal{C})}$ without loss of generality by Observation 3.1.

We claim that for any binary vector $x = (x_1, x_2, \ldots, x_n) \in \{0,1\}^n$,

$$\langle Ax, s \rangle = \sum_{i \,:\, x_i=1} z_i. \tag{2}$$

To see this, observe that

$$\begin{aligned}
\langle Ax, s \rangle &= \sum_{i \in [d]} y_i \sum_{x_j=1} A[i, j] \\
&= \sum_{x_j=1} y_1 A[1, j] + y_2 A[2, j] + \ldots y_d A[d, j] \\
&= \sum_{x_j=1} \langle y, v(z_j) \rangle \\
&= \sum_{j \,:\, x_j=1} z_j.
\end{aligned}$$

Thus $\langle Ax, s \rangle = t$ if and only if $\sum_{i \,:\, x_i=1} z_i = t$, and there is a one-to-one correspondence between solutions to our $\mathcal{C}$-Subset Sum instance and our HBILP feasibility instance. $\qquad\square$

5.2 Equivalence Between HBILP Feasibility and Subset Sum

Theorem 1.2. *$\mathcal{C}$-Subset Sum can be solved in time $O_{\mathcal{C}}(\mathsf{poly}(n))$ if and only if Hyperplane-Constrained Binary ILP (HBILP) can be solved in time $\Delta^{O(m)} \cdot O_m(\mathsf{poly}(|\mathcal{I}|))$, where $|\mathcal{I}|$ is the size of the instance.*

Theorem 1.2 follows immediately from the next two lemmas, which show reductions in both directions. The first is a consequence of the reduction in Section 5.1:

Lemma 5.2. *If HBILP Feasibility can be solved in time $\Delta^{O(m)} \cdot O_m(\mathsf{poly}(n))$, then $\mathcal{C}$-Subset Sum can be solved in time $O_{\mathcal{C}}(\mathsf{poly}(n))$ with success probability $1 - n^{-\gamma}$ for an arbitrarily large constant $\gamma > 0$.*

Proof. In polynomial time (in the size of the input), we can preprocess an instance of Subset Sum and produce an equivalent one such that all integers are bounded by $2^{\mathsf{poly}(n)}$ (see, e.g., [FT87, HN10]).[8] Next, we use the reduction given in Lemma 5.1, which takes time $\widetilde{O}_{\mathcal{C}}(n)$ and succeeds with probability $1 - n^{-\gamma}$, and solve the resulting HBILP instance in time

$$\Delta^{O(m)} \cdot O_m(\mathsf{poly}(n)) = (n^{2/d(\mathcal{C})})^{O(d(\mathcal{C}))} \cdot O_{\mathcal{C}}(\mathsf{poly}(n)) = O_{\mathcal{C}}(\mathsf{poly}(n)). \qquad\square$$

---

[8]See discussion about the computational model in Section 2.

Lemma 5.3. *If $\mathcal{C}$-Subset Sum admits an $O_{\mathcal{C}}(\mathsf{poly}(n))$-time algorithm, HBILP Feasibility can be solved in time $\Delta^{O(m)} \cdot O_m(\mathsf{poly}(n))$.*

Proof. Fix an instance of HBILP Feasibility given by the matrix $A \in \mathbb{Z}^{m \times n}$, the vector $s \in \mathbb{Z}^m$, and the integer target t. Let $\Delta := \|A\|_\infty$.

We perform the reduction in two steps. First, we self-reduce our HBILP instance to another HBILP instance A', s', t' with the property that every column $A'[\cdot, j]$ of A' has a unique dot product $\langle A'[\cdot, j], s' \rangle$. We then reduce A', s', t' to $\mathcal{C}$-Subset Sum.

If A contains any column with only zeroes, then the value of the corresponding entry of x does not matter, and we can safely delete it. Thus we can assume without loss of generality that each column of A has at least one nonzero entry. Moreover, by Observation B.2, proved in Appendix B, we can assume that each entry of A is non-negative and that any solution vector x has fixed support exactly q for some $q = \Theta(n)$.

Step 1: Self-reduction. In order to construct the instance A', s', t', define $M := nm\Delta\|s\|_\infty + 1$, which satisfies

$$M > \langle Ax, s \rangle \tag{3}$$

for any $x \in \{0, 1\}^n$ by construction. Moreover, let $k := \lceil \log_\Delta(n) \rceil$.

Let $R \in [\Delta]^{k \times n}$ be the matrix whose columns are vectors in $[0 : \Delta - 1]^k$ in lexicographically increasing order. Because the number of such vectors is at least n, every column of R is different. Recall that $J_{k \times n}$ denotes the $k \times n$ matrix containing only 1's and let $\overline{R}$ be $\Delta \cdot J_{k \times n} - R$.

Create the block matrix $A' \in \mathbb{Z}_{\geq 0}^{(m+k) \times 2n}$ as follows. The top-left block is A, the bottom-left block is R, the bottom-right block is $\overline{R}$ and each entry in the top-right block is 0. Observe that every column in $\widetilde{A}$ is distinct because each column in R is distinct and no column in A is all 0's.

Create $s' \in \mathbb{Z}_{\geq 0}^{m+k}$ as follows. The first m entries of s' are s, and the remaining k entries are the vector $v = (M\Delta^0, M\Delta^1, \ldots, M\Delta^{k-1})$. Finally, set $t' := t + q\Delta\|v\|_1$ to complete the reduction.

$$A' := \left(\begin{array}{c|c} A & \mathbf{0} \\ \hline R & \overline{R} \end{array} \right) \quad s' := \left(\begin{array}{c} s \\ \hline v \end{array} \right)$$

Claim 5.1. *For every distinct pair of indices $i, j \in [2n]$, $\langle A'[\cdot, i], s' \rangle \neq \langle A'[\cdot, j], s' \rangle$.*

Proof: Begin with the first n columns. For all $i \in [n]$, we can break down the relevant dot product into two pieces corresponding to the top and bottom portions of A':

$$\langle A'[\cdot, i], s' \rangle = \langle A[\cdot, i], s \rangle + \langle R[\cdot, i] \cdot v \rangle.$$

First, observe that $\langle R[\cdot, i], v \rangle$ is distinct for every $i \in [n]$ by construction, due to the fact that each component of $R[\cdot, i]$ is less than Δ, and the components of v increase by factors of Δ.

Second, because

$$0 < \langle A[\cdot, i], s \rangle \leq M,$$

the $\langle A[\cdot, i], s \rangle$ term of the dot product $\langle A'[\cdot, i], s' \rangle$ is not large enough to interfere with the $\langle R[\cdot, i], v \rangle$ term, and thus the first n columns of A' have distinct dot products with s'.

Because $\overline{R} = \Delta \cdot J_{k \times n} - R$, and because no column of A consists of all 0's by assumption, similar arguments show that the value $\langle A'[\cdot, i], s' \rangle$ is distinct for every column $i \in [2n]$. ∎

Claim 5.2. *The ILP instance (A', s', t') has a solution if and only if the instance (A, s, t) has a solution (and the solution to A, s, t can be recovered efficiently from the solution of A', s', t').*

Proof: Suppose x satisfies $\langle Ax, s \rangle = t$. Recall that x has support exactly q by Observation B.2 without loss of generality. Thus the vector x' created by concatenating two copies of x satisfies

$$\langle A'x', s' \rangle = t + q\Delta\|v\|_1 = t'.$$

Moreover, any vector $y' \in \{0, 1\}^{2n}$ that satisfies $\langle A'y', s' \rangle = t'$ must satisfy

$$\langle A(y'_1, y'_2, \ldots, y'_n), s \rangle = t$$

by construction. This is because the $[R \mid \overline{R}]$ submatrix of A' can contribute to t' only in multiples of M, so because $\langle A(y'_1, y'_2, \ldots, y'_n), s \rangle < M$ by (3), this product must evaluate to t. ∎

Step 2: Reduction to $\mathcal{C}$-Subset Sum. Consider the integer vector

$$z := (\langle s', A'[\cdot, 1]\rangle, \langle s', A'[\cdot, 2]\rangle, \ldots, \langle s', A'[\cdot, 2n]\rangle) \tag{4}$$

and let

$$Z = \{z_1, z_2, \ldots, z_{2n}\}$$

denote the set containing the components of z. (Note that Z is a proper set and contains no duplicates, by Claim 5.1.) We proceed to consider Z, t as an instance of Subset Sum.

Because $\langle A'x, s' \rangle = t'$ if and only if $\langle x, z \rangle = t'$ by construction (4), we have a one-to-one correspondence between solutions to our Subset Sum and HBILP Feasibility instances: any subset of Z that adds to t' corresponds to a binary vector $x \in \{0, 1\}^{2n}$ such that $\langle A'x, s' \rangle = t'$, which can be used to recover a solution for the original instance A, s, t by Claim 5.2. It remains to show that an $O_{\mathcal{C}}(\mathsf{poly}(n))$ algorithm for $\mathcal{C}$-Subset Sum will allow us to solve the problem in the claimed time.

We begin by bounding the doubling constant $\mathcal{C}$ of Z. By the definition of z, we have that $z_j = \langle s, A'[\cdot, j]\rangle$ for all $j \in [n]$, and thus Z is a subset of the GAP

$$Y := \{y_1 s_1 + y_2 s_2 + \cdots + y_m s_m + y_{m+1} M \ : \ -\Delta \le y_i \le \Delta, \forall i \in [m]; 0 < y_{m+1} < \Delta^{k-1}\}.$$

Note here that the dimension of Y is $m + 1$ instead of $m + k$, as we have chosen to represent the component of each $z_j \in Z$ divisible by M into a single large dimension.

We claim that we can assume

$$|Z| = \Omega(|Y|) \tag{5}$$

without loss of generality. To see this, observe that we can inflate $|Z|$ by adding up to $|Y|$ dummy elements from the translated GAP $t + Y$. Because every such element is greater than t, and each is contained in a translation of Y, we create no additional solutions and increase $|Y + Y|$ by at most a factor of 2.

We have that

$$\begin{aligned}
|Z + Z| &\le |Y + Y| \\
&\le 2^{m+1} \cdot |Y| \\
&= O_m(1) \cdot |Z|,
\end{aligned}$$

where the first line follows from the fact that $Z \subseteq Y$, the second line follows from the fact that $|Y|$ has dimension $m + 1$, and the third follows from (5).

Thus Z, t is an instance of $O_m(1)$-Subset Sum whose solutions correspond directly to solutions of our original HBILP feasibility instance. Also, $|Z| = O(|Y|) = \Delta^{O(m)+k}$. Because $\Delta^k = O(n)$ by the definition of k, an algorithm for Subset Sum that runs in time $O_{\mathcal{C}}(\mathsf{poly}(n))$ solves Z, t in time $O_m(\mathsf{poly}(\Delta^m \cdot n)) = \Delta^{O(m)} \cdot O_m(\mathsf{poly}(n))$ as claimed. □

5.2.1 Reduction from BILP Feasibility to HBILP Feasibility

An FPT algorithm for $\mathcal{C}$-Subset Sum further implies a $\Delta^{O(m)} \cdot O_m(\text{poly}(n))$ algorithm for Bounded ILP feasibility, i.e., an extension of Eisenbrand and Weismantel's improvement for Unbounded ILPs to determining feasibility for Bounded ILPs.

Corollary 5.2. *If $\mathcal{C}$-Subset Sum can be solved in $O_{\mathcal{C}}(\text{poly}(n))$, then Bounded ILPs defined by $A \in \mathbb{Z}^{m \times n}$, $b \in \mathbb{Z}^m$ with $\Delta := \|A\|_\infty$ and each variable x_i bounded by $\text{poly}(n)$ can be solved in time $\Delta^{O(m)} \cdot O_m(\text{poly}(n))$.*

Corollary 5.2 is a straightforward corollary of Lemma 5.4, which reduces ILP Feasibility with binary variables to HBILP feasibility, and the fact that ILPs with polynomially bounded variables can be reduced to binary ILPs (Remark 4.1). We defer the proof of Lemma 5.4 to Appendix B.

Lemma 5.4. *If HBILP Feasibility can be solved in time $\Delta^{O(m)} \cdot O_m(\text{poly}(n))$, Binary ILP Feasibility can be solved in time $\Delta^{O(m)} \cdot O_m(\text{poly}(n))$.*

6 Unbounded Subset Sum with Constant Doubling

$\mathcal{C}$-Unbounded Subset Sum is equivalent to an unbounded integer program with a single constraint. In this section, we prove a near-XP algorithm for $\mathcal{C}$-Unbounded Subset Sum by first using the constructive Freiman's theorem to map instances to integer programs with small coefficients, then using existing methods to find small-support solutions to the integer programs. The proof of the lemma uses techniques that are standard in the literature (see, e.g., [ES06]); nevertheless, we are not aware of a prior proof of the following statement.

Lemma 6.1 (ILP Solutions with small support). *Let $A \in \mathbb{Z}^{m \times n}$ with $\Delta := \|A\|_\infty$. In $(n\Delta)^{O(m)}$ time we can find a set $\mathcal{X} \subseteq \{0,1\}^n$ with the following property: For any target vector $b \in \mathbb{Z}^m$ corresponding to at least one solution $x \in \mathbb{Z}_{\geq 0}^n$ with $Ax = b$, there exists a small-support solution $y \in \mathbb{Z}_{\geq 0}^n$ satisfying*

$$Ay = b, \quad \text{supp}(y) \in \mathcal{X} \quad and \quad |\text{supp}(y)| \leq m \log_2(2n\Delta + 1).$$

Proof. We begin with a bound on the support of lexicographically minimal solutions that follows standard arguments.

Claim 6.1. *Let $A \in \mathbb{Z}^{m \times n}$ with $\Delta = \|A\|_\infty$, and let $y \in \mathbb{Z}_{\geq 0}^n$ be the lexicographically minimal vector such that $Ay = b$ for some $b \in \mathbb{Z}^m$. Then $|\text{supp}(y)| \leq m \log_2(2n\Delta + 1)$.*

Proof: Assume for contradiction that

$$2^{|\text{supp}(y)|} > (2n\Delta + 1)^m.$$

Because $Ax \leq (2n\Delta + 1)^m$ for any $x \in \{0,1\}^n$, there must exist two different vectors $v, w \in \{0,1\}^n$ such that (i) $\text{supp}(v), \text{supp}(w) \subseteq \text{supp}(y)$, and (ii) $Av = Aw$, by the pigeonhole principle.

Let $y_1 = y - w + v$ and $y_2 = y + w - v$. Observe that $Ay_1 = Ay_2$ and $y_1, y_2 \in \mathbb{Z}_{\geq 0}^n$ because $\text{supp}(v), \text{supp}(w) \subseteq \text{supp}(y)$. Moreover, because $v \neq w$ we have that one of y_1 or y_2 is lexicographically smaller than y, which contradicts the assumption that y is lexicographically minimal. ∎

Let $\mathcal{X} \subseteq \{0,1\}^n$ be the set of lexicographically minimal solutions to $Ax = b$ for every $b \in \mathbb{Z}_{\geq 0}^n$ with $\|b\|_\infty < n\Delta$. Clearly, $|\mathcal{X}| \leq (2n\Delta + 1)^m$ as this is the number of suitable b's. To construct $\mathcal{X}$ it remains to iterate over every $b \in \mathbb{Z}_{\geq 0}^n$ with $\|b\|_\infty < n\Delta$ and solve the following Integer Linear Program:

$$\max \left\{ \sum_{i=1}^n x_i \cdot M^i \mid Ax = b, x \in \mathbb{Z}_{\geq 0}^n \right\},$$

where $M = 4n\Delta$. Note that this can be solved in $(n\Delta)^{O(m)}$ time by [EW19, Theorem 2.3] for each b. Hence, the set $\mathcal{X}$ can be constructed in the claimed time. Finally, it remains to show that for any feasible b, there exists a solution y with small support in $\mathcal{X}$.

Claim 6.2. *Let $b \in \mathbb{Z}^m$ be any vector for which there exists $x \in \mathbb{Z}_{\geq 0}^n$ with $Ax = b$. Then there also exists $y \in \mathbb{Z}_{\geq 0}^n$ such that $Ay = b$ and $\mathrm{supp}(y) \in \mathcal{X}$.*

Proof: Let $z \in \mathbb{Z}_{\geq 0}^n$ be the lexicographically minimum vector such that $Az = b$. Let $\widehat{z} \in \{0,1\}^n$ be such that $\widehat{z}_i = 1$ iff $z_i \neq 0$ and $\widehat{z}_i = 0$ otherwise. Let $\widehat{b}$ be such that $A\widehat{z} = \widehat{b}$. Observe that $\|\widehat{b}\|_\infty < n\Delta$.

Hence it remains to show that $\widehat{z}$ is the lexicographically minimal vector for which $A\widehat{z} = \widehat{b}$. Assume for contradiction that there exists $\widehat{y} \in \mathbb{Z}_{\geq 0}^n$ such that $A\widehat{y} = \widehat{b}$ and $\widehat{y}$ is lexicographically smaller than $\widehat{z}$. Consider a vector $y = z - \widehat{z} + \widehat{y}$. Note, that $\mathrm{supp}(\widehat{z}) \subseteq \mathrm{supp}(z)$ so $y \in \mathbb{Z}_{\geq 0}^n$. Clearly $Ay = Az = b$. Moreover, because $\widehat{y}$ is lexicographically smaller than $\widehat{z}$, it follows that y is lexicographically smaller than z. This contradicts our assumption that z is lexicographically minimal. ∎ Thus the set $\mathcal{X}$ satisfies the property stated in Lemma 6.1, concluding the proof. □

With Lemma 6.1 in hand, let us present our algorithm for $\mathcal{C}$-Unbounded Subset Sum.

Theorem 6.1 (Near-XP algorithm for $\mathcal{C}$-Unbounded Subset Sum). *$\mathcal{C}$-Unbounded Subset Sum can be solved in time $n^{O_{\mathcal{C}}(1)}$ if an ILP instance $\mathcal{I}$ on v variables can be solved in time $2^{O(v)}\mathsf{poly}(|\mathcal{I}|)$.*

Using the current best algorithm [RR23], $\mathcal{C}$-Unbounded Subset Sum can be solved in $n^{O_{\mathcal{C}}(1)\log\log\log(n)}$ time.

Proof. Following the steps of our reduction from $\mathcal{C}$-Subset Sum to HBILP feasibility (Lemma 5.1), we can use the constructive Freiman's theorem[9] (Theorem 3.1) to encode the $\mathcal{C}$-Unbounded Subset Sum instance as an *Unbounded* Hyperplane-Constrained ILP Feasibility instance given by $A \in \mathbb{Z}_{\geq 0}^{d(\mathcal{C}) \times n}$ with $\Delta = \|A\|_\infty = n^{O(1/d(\mathcal{C}))}$, step vector ℓ, and target t.

We then use Lemma 6.1 to construct a set $\mathcal{X}$ of candidate supports in $(n\Delta)^{O(m)} = n^{O_{\mathcal{C}}(1)}$ time. For each support vector $x^* \in \mathcal{X}$, we reduce the ILP to variables in x^*. This gives us a program with $|x^*| = O(m\log_2(n\Delta)) = O_{\mathcal{C}}(\log(n))$ variables. Now, we encode this problem as the ILP

$$\left\{ x \in \mathbb{Z}_{\geq 0}^n \;\middle|\; \sum_{j=1}^d \ell_j \sum_{i \in x^*} a_{i,j} x_i = t \right\}.$$

Observe that this is equivalent to an instance of Unbounded Subset Sum with $O_{\mathcal{C}}(\log(n))$ items. Thus any algorithm for Unbounded Subset Sum (or, more generally, any algorithm for unbounded ILP) that runs in time $2^{O(v)}$ on instances with v variables would automatically yield an $n^{O_{\mathcal{C}}(1)}$ time algorithm for $\mathcal{C}$-Unbounded Subset Sum. Using the best-known algorithm for unbounded ILPs, which runs in time $(\log v)^{O(v)}$ [RR23], we get an $n^{O_{\mathcal{C}}(1)\log\log\log(n)}$-time algorithm. □

7 k-SUM with Constant Doubling

Our final contribution concerns the analogous problem of k-SUM with bounded doubling constant, which we refer to as $(\mathcal{C}, k)$-SUM. We prove Theorem 7.1 and observe that the same approach gives an algorithm for 4-SUM that is tight up to subpolynomial factors, assuming the k-SUM conjecture.

Problem 4: $(\mathcal{C}, k)$-SUM

In: An integer set $X = \{x_1, x_2, \ldots, x_n\}$ such that $|X + X| \leq \mathcal{C}|X|$ and an integer target t.
Out: $S \subseteq X$ with $|S| = k$ such that $\Sigma(S) = t$, or 'NO' if no solution exists.

We note that [ABF23] and [JX23] also present algorithms for 3-SUM in cases where additive structure in the input is controlled by the doubling constant, and also make use of fast algorithms for sparse

---

[9]We remark that because the construction of the GAP guaranteed by Freiman's theorem is not the runtime bottleneck, a slower constructive algorithm might suffice for this step.

convolution. In both cases, these authors focus on the setting of tripartite 3-SUM under the condition that at least one of the three input sets A, B, and C is guaranteed to have a small doubling.

Before we continue, let us recall the standard "color-coding" technique that allows us to ensure that each integer in the solution is taken at most once.

Lemma 7.1. *Let A be a set of n integers. There exists a set family*

$$\mathcal{P} \subseteq \{(A_1, \ldots, A_k) \text{ such that } A_1 \uplus \ldots \uplus A_k \text{ is partition of } A\}$$

with the following properties:

1. *For any $S \subseteq A$ of cardinality $|S| = k$, there exists $(A_1, \ldots, A_k) \in \mathcal{P}$ with $|S \cap A_i| = 1$ for all $i \in [k]$.*

2. *$|\mathcal{P}| = 2^{O(k)} \cdot \log(n)$.*

$\mathcal{P}$ can be constructed deterministically in $2^{O(k)} n \log(n)$ time.

The proof of Lemma 7.1 is a reformulation of a standard construction of an (n, k)-perfect hash family (see [AYZ95], Section 4). For completeness, we include a standalone proof in Appendix C. We now commence with the proof of Theorem 7.1.

Theorem 7.1. *Given an integer set X such that $|X + X| \leq \mathcal{C} \cdot |X|$ and an integer t, we can decide if there exists a set $\{x_1, \ldots, x_k\} \subseteq X$ such that $x_1 + \ldots + x_k = t$ in deterministic time $\widetilde{O}(\mathcal{C}^{\lceil k/2 \rceil} \cdot 2^{O(k)} \cdot n)$.*

Proof. Let X be an integer set of size n, and let $\{x_1, \ldots, x_k\} \subseteq X$ denote a set of k integers that sum to t. We commence by constructing the family $\mathcal{P}$ from Lemma 7.1 and guessing a partition $(X_1, \ldots, X_k) \in \mathcal{P}$ of X such that $x_i \in X_i$ for all $i \in [k]$. Observe that by Lemma 7.1 this incurs only an additional $2^{O(k)} \log(n)$ factor in the running time.

Now, we use the sparse convolution algorithm of Bringmann et al. [BFN22].

Lemma 7.2 (Theorem 1 in [BFN22]). *Given two sets $A, B \subseteq [\Delta]$, the set $A + B := \{a + b \mid a \in A, b \in B\}$ can be constructed deterministically in $\widetilde{O}(|A + B| \cdot \mathsf{polylog}(\Delta))$ time.*

We use Lemma 7.2 to enumerate two sets:

$$\mathcal{L} := X_1 + \ldots + X_{\lfloor k/2 \rfloor} \qquad \text{and} \qquad \mathcal{R} := X_{\lfloor k/2 \rfloor + 1} + \ldots + X_k.$$

Both $\mathcal{L}$ and $\mathcal{R}$ can be computed deterministically in $\widetilde{O}(k \cdot (|\mathcal{L}| + |\mathcal{R}|))$ time by repeatedly applying Lemma 7.2. Next, with both $\mathcal{L}$ and $\mathcal{R}$ in hand, we apply the meet-in-the-middle approach to recover a solution if one exists. This can be implemented in $\widetilde{O}(|\mathcal{L}| + |\mathcal{R}|)$ time by first sorting $\mathcal{L}$ and $\mathcal{R}$, and then for every element $a \in \mathcal{L}$ using binary search to decide if $t - a \in \mathcal{R}$. Finally, if for at least one $a \in \mathcal{L}$ we find an accompanying element in $\mathcal{R}$, we know that the instance has a solution.

As stated, the algorithm decides k-SUM without recovering a solution. However, given that a solution exists we can recover a solution via binary search at the cost of an additional $O_k(\log(n))$ factor. This concludes the description of the algorithm.

Correctness of the algorithm follows from the fact that Lemma 7.1 returns a valid partition, and from the definition of the sets $\mathcal{L}$ and $\mathcal{R}$. It remains to bound the runtime. Since the other steps of the algorithm take time $\widetilde{O}_k(n)$, the bottleneck occurs in the meet-in-the-middle step, which takes time $\widetilde{O}_k(|\mathcal{L}| + |\mathcal{R}|)$. Therefore it remains to bound the sizes of $\mathcal{L}$ and $\mathcal{R}$. Without loss of generality, consider $|\mathcal{R}|$, and observe

$$|\mathcal{R}| = |X_1 + \ldots + X_{\lceil k/2 \rceil}| \leq |\lceil k/2 \rceil X| = \mathcal{C}^{\lceil k/2 \rceil} |X|,$$

where the final step applies Plünnecke-Ruzsa (Lemma A.1). This concludes the proof of Theorem 7.1. $\qquad \square$

In the specific case of $k = 4$, using the doubling constant directly gives a slightly better bound. The resulting algorithm for $(\mathcal{C}, 4)$-SUM is optimal up to subpolynomial factors under the 4-SUM conjecture.

Corollary 7.1. $(\mathcal{C}, 4)$-*SUM can be solved in expected time* $\widetilde{O}(\mathcal{C}n)$. *Moreover, for any constant* $\varepsilon > 0$, $(\mathcal{C}, 4)$-*SUM cannot be solved in* $O(\mathcal{C}^{1-\varepsilon}n)$ *time unless* 4-*SUM can be solved in time* $O(n^{2-\varepsilon})$ *for* $\varepsilon > 0$.

Proof. The upper bound follows by analysis of the proof of Theorem 7.1. Recall that the bottleneck is $|\mathcal{R}|$, which in the case when $k = 4$ is $|X + X| \leq \mathcal{C} \cdot |X|$.

For the lower bound, observe that $|X + X| \leq |X|^2$ and therefore $\mathcal{C} \leq |X|$. Thus, any algorithm for $(\mathcal{C}, 4)$-SUM with runtime $O(\mathcal{C}^{1-\varepsilon}n)$ would yield an algorithm for 4-SUM that runs in $O(n^{2-\varepsilon})$ time. $\square$

Remark 7.1. Applying the same lower bound argument to the more general case of k-SUM gives a lower bound of $\Omega(\mathcal{C}^{\lceil k/2 \rceil - 1}n)$ for $(\mathcal{C}, k)$-SUM under the k-SUM conjecture, leaving an $O(\mathcal{C})$-factor gap.

In [Pet11], Petridis gives the slightly improved bound

$$|hA| = O(\mathcal{C}^{h-1}|A|^{2-\frac{1}{h}}),$$

for finite sets in commutative groups, improving on Plünnecke-Ruzsa for our purposes. This narrows the gap between our upper and lower bounds slightly, although the result is still not tight for $k > 4$. Further improvements, perhaps by non-trivially leveraging the small doubling constant of the input set to achieve a better algorithmic result for large k, would be both interesting and surprising.

Acknowledgements

References

[ABF23]   Amir Abboud, Karl Bringmann, and Nick Fischer. Stronger 3-SUM Lower Bounds for Approximate Distance Oracles via Additive Combinatorics. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 391–404. ACM, 2023. 1.1, 1.1, 3, 7, A.4, A.5

[ABHS22] Amir Abboud, Karl Bringmann, Danny Hermelin, and Dvir Shabtay. SETH-based Lower Bounds for Subset Sum and Bicriteria Path. *ACM Transactions on Algorithms (TALG)*, 18(1):1–22, 2022. 1.1

[Agr04]   Agrawal, Manindra and Kayal, Neeraj and Saxena, Nitin. PRIMES is in P. *Annals of mathematics*, pages 781–793, 2004. A.1, A.5

[AKKN16] Per Austrin, Mikko Koivisto, Petteri Kaski, and Jesper Nederlof. Dense Subset Sum may be the hardest. *33rd Symposium on Theoretical Aspects of Computer Science (STACS 2016)*, pages 13:1–13:14, 2016. 1.1

[AYZ95]   Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *Journal of the ACM (JACM)*, 42(4):844–856, 1995. 7

[BFN22]   Karl Bringmann, Nick Fischer, and Vasileios Nakos. Deterministic and Las Vegas Algorithms for Sparse Nonnegative Convolution. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3069–3090. SIAM, 2022. 1.2, 7, 7.2

[Bib13]   Khodakhast Bibak. Additive Combinatorics: With a View Towards Computer Science and Cryptography—An Exposition. In *Number Theory and Related Fields*, pages 99–128, New York, NY, 2013. Springer New York. 1.1

[Bri17]   Karl Bringmann. A Near-Linear Pseudopolynomial Time Algorithm for Subset Sum. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1073–1084. SIAM, 2017. 1.1

[BW21]   Karl Bringmann and Philip Wellnitz. On Near-Linear-Time Algorithms for Dense Subset Sum. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021*, pages 1777–1796. SIAM, 2021. 1.1

[CEH+21]   Jana Cslovjecsek, Friedrich Eisenbrand, Christoph Hunkenschröder, Lars Rohwedder, and Robert Weismantel. Block-Structured Integer and Linear Programming in Strongly Polynomial and Near Linear Time. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021*, pages 1666–1681. SIAM, 2021. 1.1

[CEP+21]   Jana Cslovjecsek, Friedrich Eisenbrand, Michał Pilipczuk, Moritz Venzin, and Robert Weismantel. Efficient Sequential and Parallel Algorithms for Multistage Stochastic Integer Programming Using Proximity. In *29th Annual European Symposium on Algorithms, ESA 2021*, volume 204 of *LIPIcs*, pages 33:1–33:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. 1.1

[Cha02]   Mei-Chu Chang. A Polynomial Bound in Freiman's Theorem. *Duke Math. J.*, 115(1):399–419, 2002. A

[CJRS23]   Xi Chen, Yaonan Jin, Tim Randolph, and Rocco A. Servedio. Subset Sum in Time $2^{n/2}/poly(n)$, 2023. 1.1

[CKL+24]   Jana Cslovjecsek, Martin Koutecký, Alexandra Lassota, Michał Pilipczuk, and Adam Polak. Parameterized algorithms for block-structured integer programs with large entries. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 740–751. SIAM, 2024. 1.1

[CL15]   Timothy M Chan and Moshe Lewenstein. Clustered Integer 3SUM via Additive Combinatorics. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 31–40, 2015. 1.1

[CLMZ23]   Lin Chen, Jiayi Lian, Yuchen Mao, and Guochuan Zhang. Faster Algorithms for Bounded Knapsack and Bounded Subset Sum Via Fine-Grained Proximity Results. *CoRR*, abs/2307.12582, 2023. 1.1

[DLRV23]   Daniel Dadush, Arthur Léonard, Lars Rohwedder, and José Verschae. Optimizing Low Dimensional Functions over the Integers. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 115–126. Springer, 2023. 1.2, 5.1

[ES06]   Friedrich Eisenbrand and Gennady Shmonin. Carathéodory bounds for integer cones. *Oper. Res. Lett.*, 34(5):564–568, 2006. 6

[EW19]   Friedrich Eisenbrand and Robert Weismantel. Proximity Results and Faster Algorithms for Integer Programming Using the Steinitz Lemma. *ACM Transactions on Algorithms (TALG)*, 16(1):1–14, 2019. 1.1, 1.1, 1.2, 6

[Fre64]   Gregory A Freiman. On the addition of finite sets. In *Doklady Akademii Nauk*, volume 158, pages 1038–1041. Russian Academy of Sciences, 1964. 3.1

[FT87]   András Frank and Éva Tardos. An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica*, 7:49–65, 1987. 5.2

[HN10]   Danny Harnik and Moni Naor. On the Compressibility of NP Instances and Cryptographic Applications. *SIAM Journal on Computing*, 39(5):1667–1713, 2010. 5.2

[HS74]   Ellis Horowitz and Sartaj Sahni. Computing partitions with applications to the knapsack problem. *Journal of the ACM (JACM)*, 21(2):277–292, 1974. 1.1

[JR18]   Klaus Jansen and Lars Rohwedder. On integer programming and convolution. In *10th Innovations in Theoretical Computer Science Conference (ITCS 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018. 1.1, 1.2

[JX23]   Ce Jin and Yinzhan Xu. Removing Additive Structure in 3SUM-Based Reductions. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 405–418, 2023. 1.1, 1.1, 7

[KKM20]   Dušan Knop, Martin Koutecký, and Matthias Mnich. Combinatorial n-fold integer programming and applications. *Math. Program.*, 184(1):1–34, 2020. 1.1

[KPW20]   Dušan Knop, Michał Pilipczuk, and Marcin Wrochna. Tight complexity lower bounds for integer linear programming with few constraints. *ACM Transactions on Computation Theory (TOCT)*, 12(3):1–19, 2020. 1.2

[KX19]   Konstantinos Koiliaris and Chao Xu. Faster pseudopolynomial time algorithms for subset sum. *ACM Transactions on Algorithms (TALG)*, 15(3):1–20, 2019. 1.1

[Len83]   Henrik W. Lenstra, Jr. Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8(4):538–548, 1983. 1.1

[Lov17]   Shachar Lovett. *Additive Combinatorics and its Applications in Theoretical Computer Science*. Number 8 in Graduate Surveys. Theory of Computing Library, 2017. 1.1

[NSS95]   Moni Naor, Leonard J Schulman, and Aravind Srinivasan. Splitters and near-optimal derandomization. In *Proceedings of IEEE 36th Annual Foundations of Computer Science*, pages 182–191. IEEE, 1995. C, C.1

[NW21]   Jesper Nederlof and Karol Węgrzycki. Improving Schroeppel and Shamir's Algorithm for Subset Sum via Orthogonal Vectors. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1670–1683. ACM, 2021. 1.1

[Pap81]   Christos H. Papadimitriou. On the complexity of integer programming. *J. ACM*, 28(4):765–768, 1981. 1.1

[Pet11]   Giorgis Petridis. Upper bounds on the cardinality of higher sumsets. *Acta Arithmetica*, 158, 01 2011. 7.1

[PRW21]   Adam Polak, Lars Rohwedder, and Karol Węgrzycki. Knapsack and Subset Sum with Small Items. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021*, volume 198 of *LIPIcs*, pages 106:1–106:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. 1.1

[RR23]   Victor Reis and Thomas Rothvoss. The subspace flatness conjecture and faster integer programming. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 974–988. IEEE, 2023. 1.1, 6.1, 6

[Ruz94]   Imre Z. Ruzsa. Generalized arithmetical progressions and sumsets. *Acta Mathematica Hungarica*, 65(4):379–388, 1994. A

[Ruz09]   Imre Z Ruzsa. Sumsets and structure. *Combinatorial number theory and additive group theory*, pages 87–210, 2009. 4.1

[San12]   Tom Sanders. On the Bogolyubov–Ruzsa lemma. *Analysis & PDE*, 5(3):627–655, 2012. A

[San13]   Tom Sanders. The structure theory of set addition revisited. *Bulletin of the American Mathematical Society*, 50(1):93–127, 2013. A

[Sch11]   Tomasz Schoen. Near optimal bounds in Freiman's theorem. *Duke Mathematical Journal*, 158(1):1–12, 2011. A

[Tre09]   Luca Trevisan. Additive Combinatorics and Theoretical Computer Science. *ACM SIGACT News*, 40:50–66, 2009. 1.1

[TV06]   Terence Tao and Van H Vu. *Additive Combinatorics*, volume 105. Cambridge University Press, 2006. 1

[Vio11]   Emanuele Viola. *Selected Results in Additive Combinatorics: An Exposition*. Number 3 in Graduate Surveys. Theory of Computing Library, 2011. 1.1

[Woe08]   Gerhard J. Woeginger. Open problems around exact algorithms. *Discrete Applied Mathematics*, 156(3):397–405, 2008. 1.1

[Zha22]   Yufei Zhao. Graph theory and additive combinatorics. *Notes for MIT*, 18:49–58, 2022. 3.1, A.2, A.1, A.3, A.2, A.4, A.3, A.5

A   Proof of Theorem 3.2

Making Freiman's Theorem constructive is not difficult from a strictly algorithmic perspective. However, verifying the result requires close attention to the structure of the original proof and requires concepts from additive combinatorics, group theory and the geometry of numbers along the way. For this reason, we closely follow Zhao's recent exposition of a proof due to Ruzsa [Ruz94], making modifications where necessary. We wish to emphasize that *neither the existential results nor the overall proof structure below are novel*. Our contribution is the introduction of algorithmic techniques required to make the proof constructive in near-linear time.

Given a set X of cardinality n, our proof constructs a GAP P of dimension $d(\mathcal{C}) = 2^{\mathcal{C}^{O(1)}}$ and volume $v(\mathcal{C}) = 2^{2^{\mathcal{C}^{O(1)}}} n$, as in the original statement of Freiman's Theorem. We do not attempt to optimize these functions, but we suspect that techniques used to optimize d and v in subsequent proofs of Freiman's Theorem (e.g., [Cha02, Sch11, San12, San13]) could be used to improve the dependence on $\mathcal{C}$ in our results.

At several points, we make use of the Plünnecke-Ruzsa Inequality, a useful bound on the additive "growth rate" of integer sets with a small doubling constant:

Lemma A.1 (Plünnecke-Ruzsa Inequality)**.** *If X is a finite subset of an abelian group and $|X + X| \leq \mathcal{C}|X|$ for a constant $\mathcal{C}$, then for all nonnegative integers s and t,*

$$|sX - tX| \leq \mathcal{C}^{s+t}|X|.$$

A.1   Ruzsa's Modeling Lemma

A core ingredient in Freiman's theorem is Ruzsa's Modeling Lemma. This allows us to take an integer set A and map a large piece of it to small, finite group (specifically, the prime cyclic group $\mathbb{Z}_q$) in such a way that additive structure is "preserved": the image in $\mathbb{Z}_q$ behaves isomorphically to the preimage in $\mathbb{Z}$ under addition, up to a certain fixed number s of additions. The size q of the prime cyclic group is

controlled by the size of $|sA - sA|$, which is related to the doubling constant by the Plünnecke-Ruzsa Inequality.

A map that preserves additive structure in this way is called a *Freiman s-isomorphism*:

Definition A.1 (Freiman Homomorphism and Isomorphism). Given subsets A and B of two (possibly different) abelian groups and a positive integer $s \geq 2$, $\phi : A \to B$ is a *Freiman s-homomorphism* if

$$\phi(a_1) + \cdots + \phi(a_s) = \phi(a_1') + \cdots + \phi(a_s')$$

for all pairs of s-tuples in A satisfying $a_1 + \cdots + a_s = a_1' + \cdots + a_s'$. ϕ is a *Freiman s-isomorphism* if ϕ is a bijection and both ϕ and ϕ^{-1} are Freiman s-homomorphisms.

Lemma A.2 (Constructive Ruzsa's Modeling Lemma, c.f. [Zha22] Theorem 7.7.3). *Let A be a set of n integers with $|A + A| \leq \mathcal{C}|A|$, set $\Delta = \max_{a \in A} |a|$, let $s \geq 2$ be a fixed constant, and set $m = 4\mathcal{C}^{2s}n$. There exists an $O(n + \mathsf{polylog}(\Delta))$-time algorithm that:*

1. *with probability at least $1/2$, returns a mapping $\psi : \mathbb{Z} \to \mathbb{Z}_m$ and a set $A' \subset A$ with $|A'| \geq |A|/s$ such that ψ is a s-Freiman isomorphism from A' to $\psi(A')$, and*

2. *with probability at most $1/2$, returns 'failure'.*

Proof. Fix any prime $q > \max(sA - sA)$. As $|\max(sA - sA)| \leq s\Delta$, we can find a prime of this size with high probability in time $O_s(\mathsf{polylog}(\Delta))$ by repeatedly guessing and testing primality [Agr04].

For each value $\lambda \in [q]$, let ϕ_λ denote the map $\phi_\lambda : \mathbb{Z} \to \mathbb{Z}_q \to \mathbb{Z}_q \to \mathbb{Z}$ that

1. first maps $a \in \mathbb{Z}$ to $a' = a \pmod{q} \in \mathbb{Z}_q$,

2. computes $a'' = \lambda a'$ in $\mathbb{Z}_q$,

3. then maps a'' back to $[0 : q - 1] \subset \mathbb{Z}$ via the identity map.

Choose $\boldsymbol{\lambda} \in [q-1]$ uniformly at random. Since q is prime, any element $r \in [q-1]$ is a generator for $\mathbb{Z}_q$, and thus for any $r \in [q-1]$, $\phi_{\boldsymbol{\lambda}}(r)$ is uniformly distributed over $[q-1]$. Since $q > \max(sA - sA)$, for any nonzero integer $c \in sA - sA$, $c \in [q-1]$ and thus $\phi_{\boldsymbol{\lambda}}(c)$ is uniformly random over $[q-1]$.

Thus for any nonzero $c \in sA - sA$ the probability that $\phi_{\boldsymbol{\lambda}}(c)$ is divisible by m is less than $2/m$. As $m = 4\mathcal{C}^{2s}n \geq 4|sA - sA|$ by Lemma A.1, $|sA - sA| \leq m/4$ and the probability that m evenly divides *any* nonzero element in $sA - sA$ is less than $1/2$ by a union bound. Compute $\phi_{\boldsymbol{\lambda}}(A)$ in time $O(n)$ and output "failure" if m divides any element in this set. Otherwise, continue.

Let A' be a subset of A such that $|A'| \geq n/s$ and $\mathrm{diam}(\phi_{\boldsymbol{\lambda}}(A')) \leq q/s$. Note that the existence of A' is guaranteed by the pigeonhole principle. We can compute A' in time $O(n)$ by partitioning $[q]$ into evenly-sized intervals and computing $\phi_{\boldsymbol{\lambda}}(A)$.

Finally, we define $\psi_{\boldsymbol{\lambda}} : \mathbb{Z} \to \mathbb{Z}_m$ by $\psi_{\boldsymbol{\lambda}}(x) = \phi_{\boldsymbol{\lambda}}(x) \pmod{m}$ and observe that $\psi_{\boldsymbol{\lambda}}$ is a s-isomorphism from A' to $\psi_{\boldsymbol{\lambda}}(A')$ as m does not divide any nonzero element in $sA - sA$. This follows from the final two paragraphs of the proof of Theorem 7.7.3 in [Zha22], with the argument unchanged. $\square$

A.2 Bogolyubov's Lemma in $\mathbb{Z}_m$

Given a relatively large set $B \in \mathbb{Z}_m$, Bogolyubov's Lemma states that $2B - 2B$ contains a set of points that behaves "like a subspace" in the sense that each point is "close to orthogonal" to a certain set $R \in \mathbb{Z}_m$. Specifically, we employ the concept of a *Bohr set*, defined as

$$\mathrm{Bohr}_m(R, \varepsilon) := \{x \in \mathbb{Z}_m : \|rx/m\|_{\mathbb{R}/\mathbb{Z}} \leq \varepsilon, \text{ for all } r \in R\}.$$

(Recall that the norm $\|\cdot\|_{\mathbb{R}/\mathbb{Z}}$ denotes distance from the nearest integer.) We refer to $|R|$ as the *dimension* and ε as the *width* of the Bohr set.

A Bohr set is analogous to a subspace of codimension $|R|$, in the sense that if we add together several elements of a Bohr set, their sum is still close to a multiple of m when scaled by any $r \in R$. In this sense, we can view Bogolyubov's lemma as a statement that sets of the form $2B - 2B \in \mathbb{Z}_m$ contain subsets with group-like structure.

Lemma A.3 (Constructive Bogolyubov's lemma for $\mathbb{Z}_m$, c.f. [Zha22] Theorem 7.8.5). *Given $B \subseteq \mathbb{Z}_m$ with $|B| = \alpha m$, we can compute $R \subseteq \mathbb{Z}_m$ of dimension $|R| < 1/\alpha^2$ such that $Bohr(R, 1/4) \subseteq 2B - 2B$ in time $\widetilde{O}(m)$.*

Proof. To make Bogolyubov's lemma in $\mathbb{Z}_m$ [Zha22, Theorem 7.8.5] constructive, it suffices to observe that R is defined explicitly as

$$R = \{r \in \mathbb{Z}_m \setminus \{0\} \ : \ |\widehat{\mathbb{1}_B}(r)| > \alpha^{3/2}\}.$$

Here $\widehat{\mathbb{1}_B}$ is the finite group Fourier transform of $\mathbb{1}_B$, the membership function of B:

$$\widehat{\mathbb{1}_B}(r) = \frac{1}{m} \sum_{x \in \mathbb{Z}_m} \mathbb{1}_B(x) e^{-\frac{2\pi i r x}{m}}.$$

Computing R directly using the Fast Fourier Transform takes time $O(m \log m)$. $\qquad \square$

A.3   Finding a GAP in a Bohr Set

The structured nature of the Bohr set is instrumental in constructing a generalized arithmetic progression: in fact, we can show that every Bohr set contains a large GAP. In order to prove this, we need to introduce definitions from the geometry of numbers.

Definition A.2 (Successive Minima and Directional Basis). Let $\Lambda \subseteq \mathbb{R}^d$ be a lattice and $T \subseteq \mathbb{R}^d$ be a centrally symmetric convex body.

    For $i \in [d]$, the *ith successive minimum λ_i* of T with respect to Λ is the minimum value such that $\Lambda \cap \lambda_i \cdot T$ contains i linearly independent lattice vectors.

    A *directional basis* of T with respect to Λ is a basis $\{b_1, b_2, \ldots, b_d\}$ of $\mathbb{R}^d$ such that for each $i \in [d]$, $b_i \in \lambda_i T$.

    In visual terms, we can imagine constructing a directional basis by gradually scaling the convex body T outward from the origin. Every time T engulfs a new lattice vector v, we add v to our directional basis if and only if v is linearly independent from the current set of basis vectors.

Lemma A.4 (Constructing a Large GAP in a Bohr Set, c.f. [Zha22] Theorem 7.10.1). *Let m be a prime. Given a set $R \subseteq \mathbb{Z}_m$ of size $|R| = d$ and $\varepsilon \in (0, 1)$, we can compute a proper GAP $P \subseteq Bohr(R, \varepsilon)$ with dimension at most d and volume at least $(\varepsilon/d)^d m$ in time $\widetilde{O}_d(m)$.*

Proof. Let $R = \{r_1, r_2, \ldots, r_d\}$ be a subset of $\mathbb{Z}_m$ (recall that m is a prime). We can directly compute the vector $v = (\frac{r_1}{m}, \ldots, \frac{r_d}{m}) \in \mathbb{R}^d$ to define the lattice

$$\Lambda = \mathbb{Z}^d + \mathbb{Z}v \subseteq \mathbb{R}^d.$$

Note that the lattice vectors are not necessarily integral, and we have not yet computed a lattice basis; letting e_i denote the standard basis vector in dimension i, the set $\{e_1, e_2, \ldots, e_d, v\}$ spans the lattice but is not linearly independent.

    Let r' be any nonzero element of R. Since $\mathbb{Z}_m$ is a cyclic group of prime order, r' generates $\mathbb{Z}_m$. Thus, because one component of v is r'/m, the translations of the integer lattice $\{\mathbb{Z}^d + \gamma v\}_{\gamma \in [m]}$ are all disjoint. Since

$$\Lambda = \mathbb{Z}^d + \mathbb{Z}v \subseteq \bigcup_{\gamma \in [m]} \mathbb{Z}^d + \gamma v,$$

we have that Λ is the disjoint union of m translates of the integer lattice. This implies that there are exactly m lattice points of Λ within each translation of the unit cube, and, equivalently, that $\det(\Lambda) = 1/m$.

As a result, we can enumerate the set

$$C := \{\|\gamma r/m\|_{\mathbb{R} \setminus \mathbb{Z}} \; : \; \gamma \in [m]\} - \{0, 1\}^d,$$

the set of all $2^d \cdot m$ lattice points in the cube $\Lambda \cap [-1, 1)^d$, in time $O(2^d \cdot m) = O_d(m)$.

Next, we sort the set C according to the L_∞ metric, which takes time $\widetilde{O}_d(m)$. This coincides with our definition of the successive minima of a cube centered on the origin with respect to Λ: if λ_i is the ith successive minima of $[-\varepsilon, \varepsilon]^d$ with respect to Λ, then the ith directional basis vector satisfies $\|b_i\|_\infty \leq \lambda_i[-\varepsilon, \varepsilon]^d$.

Construct the successive minima $\lambda_1, \ldots, \lambda_d$ and the directional basis $b_1, \ldots, b_d$ of $[-\varepsilon, \varepsilon]^d$ with respect to Λ by greedily adding independent lattice vectors to our basis from short to long according to the L_∞ metric. Checking whether each subsequent lattice vector is independent from the previous set takes time $O_d(1)$ using Gaussian elimination. Because $[-1, 1)^d$ contains d linearly independent lattice vectors (consider the standard basis), our directional basis is guaranteed to be contained in $\Lambda \cap [-1, 1)^d = C$.

To complete the construction of the GAP in [Zha22, Theorem 7.10.1], we observe that the proper GAP P is defined explicitly in terms of the directional basis of $[-\varepsilon, \varepsilon]^d$ with respect to Λ that we have just constructed. Specifically, we have

$$P = \{\ell_1 x_1 + \cdots + \ell_d x_d \mid \forall i \in [d], \ell_i \in [L_i]\},$$

where each x_i is the unique element in $[0 : m - 1]$ such that $b_i \in x_i v + \mathbb{Z}^d$ and $L_i := \lceil 1/(\lambda_i d) \rceil$. Each L_i can be computed directly, and each x_i can be computed in time $O(m)$. $\qquad\square$

A.4 Ruzsa's Covering Lemma

Ruzsa's covering lemma states that if the sumset $|Y + Z|$ is small relative to $|Y|$, it is possible to cover Z with a small number of translates of $Y - Y$. A rough intuition for this result is that it is a statement about the "conservation of additive structure": if Y and Z have "common additive structure" (captured by the condition that $|Y + Z| \leq \mathcal{C}|Y|$), then Z is "similar" to $Y - Y$ (in the sense that Z is covered by few translates of $Y - Y$).

The fact that Ruzsa's covering lemma can be made efficiently constructive was previously observed by Abboud, Bringmann, and Fischer [ABF23]:

Lemma A.5 (Constructive Ruzsa's Covering Lemma, [ABF23] Lemma 4.7)**.** *Let Y, Z be nonempty finite subsets of an abelian group. If $|Y + Z| \leq \mathcal{C}|Y|$, then there exists a subset $X \subseteq Z$ with $|X| \leq \mathcal{C}$ and $Z \subseteq Y - Y + X$. Moreover, X can be computed in time $\widetilde{O}(\frac{|Y-Y+Z| \cdot |Y+Z|}{|Y|}) = \widetilde{O}(\mathcal{C}|Y - Y + Z|)$.*

A.5 Proof of Theorem 3.2

Proof. Combining the ingredients from the previous subsections allows us to prove Theorem 3.2. Let A be a finite integer set with $|A + A| \leq \mathcal{C}|A| = \mathcal{C}n$. By Lemma A.1, $|8A - 8A| \leq \mathcal{C}^{16}|A|$.

Choose a prime $m = O_{\mathcal{C}}(n)$ satisfying $4\mathcal{C}^{16}n < m < 16\mathcal{C}^{16}n$, which can be accomplished in time $O_{\mathcal{C}}(\text{polylog}(n))$ with high probability by guessing and testing primality [Agr04]. Then, apply Lemma A.2 with $s = 8$ to compute a set $A' \subseteq A$ with $|A'| \geq |A|/8$ and a mapping ψ such that ψ is a Freiman 8-isomorphism from A' to $B := \psi(A') \subseteq \mathbb{Z}_m$ with probability at least $1/2$ in time $\widetilde{O}(n)$. We can increase the success probability of this step by repetition: for any integer constant $\gamma > 0$, running the algorithm $\gamma \log(n)$ times lowers the failure probability to $n^{-\gamma}$.

Apply Lemma A.3 to B with

$$\alpha = \frac{|B|}{m} = \frac{|A'|}{m} \geq \frac{|A|}{8m} = 1/O_{\mathcal{C}}(1).$$

This gives us $R \subseteq \mathbb{Z}_m$ of size $1/\alpha^2 = O_\mathcal{C}(1)$ such that $Bohr(R, 1/4) \subseteq 2B - 2B$ in time $\widetilde{O}(m) = \widetilde{O}(n)$. Then, apply Lemma A.4 to R to compute the proper GAP $P \subset Bohr(R, \varepsilon) \subseteq 2B - 2B$ with dimension at most $|R| = O_\mathcal{C}(1)$ and volume at least $(1/4|R|)^{|R|}m = m/O_\mathcal{C}(1)$.

Following the proof of Theorem 7.11.1 (Freiman's Theorem) in [Zha22], we have that $Q := \psi^{-1}(P)$ is a GAP of the same dimension and volume satisfying

$$Q \subseteq 2A' - 2A' \subseteq 2A - 2A. \tag{6}$$

Thus

$$|Q + A| \leq |2A - 2A + A| = |3A - 2A| \leq \mathcal{C}^5|A| = O_\mathcal{C}(1) \cdot |Q|,$$

where the second inequality uses Lemma A.1. Using the fact that $|Q + A| = O_\mathcal{C}(1) \cdot |Q|$, we apply Lemma A.5 to Q and A to get a set X of size $|X| = O_\mathcal{C}(1)$ satisfying $A \subseteq Q - Q + X$ in time

$$\widetilde{O}(\mathcal{C}|Q - Q + A|) = \widetilde{O}(\mathcal{C}|2A - 2A - (2A - 2A) + A|) = \widetilde{O}(\mathcal{C}|5A - 4A|) = \widetilde{O}(\mathcal{C}^9|A|),$$

where the final equality uses Lemma A.1. We conclude with the observation that $Q - Q + X$ is a GAP of dimension $O_\mathcal{C}(1)$ and volume $O_\mathcal{C}(1) \cdot |Q| = O_\mathcal{C}(1) \cdot |A|$, containing A. Note that each step in the proof takes $\widetilde{O}_\mathcal{C}(n)$ time. □

A.6 Proof of Observation 3.1

Proof. Suppose $L_i > n^{1/d(\mathcal{C})}$ for some $i \in [d(\mathcal{C})]$, and let $\alpha_i := \alpha_i(\mathcal{C})$ be the solution to $L_i \leq n^{\alpha_i/d(\mathcal{C})}$. (Note that $\alpha_i = O_\mathcal{C}(1)$, as $|A| = O_\mathcal{C}(n)$.)

Let $\widehat{\alpha}_i := \alpha_i - \lfloor\alpha_i\rfloor$ denote the decimal part of α_i, and observe that

$$\{y_i\ell_i : \ell_i \in [L_i]\} \subseteq \tag{7}$$
$$\{y_{i,1}\ell_{i,1} + \cdots + y_{i,\lfloor\alpha_i\rfloor}\ell_{i,\lfloor\alpha_i\rfloor} + y_{i,\lceil\alpha_i\rceil}\ell_{i,\lceil\alpha_i\rceil} : \forall j \in [\lfloor\alpha_i\rfloor], \ell_{i,j} \in [\lceil L_i^{1/d(\mathcal{C})}\rceil], \ell_{i,\lceil\alpha_i\rceil} \in [n^{\widehat{\alpha}_i/d(\mathcal{C})}]\}, \tag{8}$$

where $y_{i,j} = y_i\lceil L_i^{1/d(\mathcal{C})}\rceil^{j-1}$ for $j \in [\lceil\alpha_i\rceil]$; that is, we can replace one dimension of our arithmetic progression with $\lceil\alpha_i\rceil$ new dimensions, each bounded by $n^{1/d(\mathcal{C})}$. As

$$vol(P) = \prod_{i \in [d]} L_i = O_\mathcal{C}(n)$$

by Theorem 3.2, performing this operation for each $L_i > n^{1/d(\mathcal{C})}$ results in a new gap P' with dimension $d'(\mathcal{C}) \leq 2d(\mathcal{C})$ and volume $O_\mathcal{C}(n)$. □

B ILP Manipulations

This appendix contains manipulations that allow us to assume certain properties of ILPs without loss of generality.

B.1 Non-negativity for BILP Feasibility

Observation B.1. Let $\mathcal{I}$ be an instance of ILP feasibility with binary variables given by the constraint matrix $A \in \mathbb{Z}^{m \times n}$ and the target vector $b \in \mathbb{Z}^m$. Without loss of generality, we can assume that entries of A are non-negative and that every solution has fixed support q for some $q = \Theta(n)$.

Proof. Construct a new constraint matrix $\widetilde{A}$ as follows: Recall that $J_{m \times n}$ denotes the m by n matrix of 1's, and add $\Delta \cdot J_{m \times n}$ to the matrix A. Then we append an additional n columns to A, where each column consists only of Δ entries only. Finally, we append a row of 1's.

To create $\widetilde{b}$, add $n\Delta \cdot J_{m \times 1}$ to b and append a single entry with the value n.

$$\widetilde{A} := \left(\begin{array}{c|c} A + \Delta \cdot J_{m \times n} & \Delta \cdot J_{m \times n} \\ \hline J_{1 \times n} & J_{1 \times n} \end{array} \right) \quad \widetilde{b} := \left(\begin{array}{c} b + n\Delta \cdot J_{m \times 1} \\ \hline n \end{array} \right)$$

Observe that every entry in $\widetilde{A}$ is positive and that the maximum entry in $\widetilde{A}$ is at most $2\Delta = O(\Delta)$. For correctness, note that the last row ensures that any solution to $\widetilde{A}x = \widetilde{b}$ with $x \in \{0,1\}^{2n}$ has support exactly n. This implies that the additional Δ factors added to every component in each of the first m rows add a total of $n\Delta$ to each component of Ax. Thus $\widetilde{A}x = \widetilde{b}$ if and only if $Ax = b$. □

B.2 Non-negativity for HBILP Feasibility

Observation B.2. Let $\mathcal{I}$ be an instance of HBILP feasibility given by the constraint matrix $A \in \mathbb{Z}^{m \times n}$, the step vector $s \in \mathbb{Z}^m$, and the target $t \in \mathbb{Z}$. Without loss of generality, we can assume that entries of A are non-negative and that every solution x has fixed support size q for some $q = \Theta(n)$.

Proof. Given A, s, t, we create a new HBILP feasibility instance $\widetilde{A}, \widetilde{s}, \widetilde{t}$ as follows. Recall that $J_{m \times n}$ denotes the m by n matrix of 1's and add $\Delta \cdot J_{m \times n}$ to A. Then append the matrix $\Delta \cdot J_{m \times n}$ to the right-hand side A. Finally, add a row of 1's to the bottom of the matrix.

Define
$$M := \|s\|_\infty \cdot 3n\Delta m,$$

and note that, by construction, we have

$$\langle (A + 2\Delta J_{m \times n})x, s \rangle \leq 3n\Delta \langle J_{m \times 1}, s \rangle < M. \tag{9}$$

Create $\widetilde{s} \in \mathbb{Z}^{m+1}$ by appending M to s, and set $\widetilde{t} = t + n\Delta\|s\|_1 + nM$. Written as block matrices, we have:

$$\widetilde{A} := \left(\begin{array}{c|c} A + \Delta \cdot J_{m \times n} & \Delta \cdot J_{m \times n} \\ \hline J_{1 \times n} & J_{1 \times n} \end{array} \right) \quad \widetilde{s} := \left(\begin{array}{c} s \\ \hline M \end{array} \right)$$

Observe that every entry in $\widetilde{A}$ is positive and that the maximum entry in $\widetilde{A}$ is at most $2\Delta = O(\Delta)$. Because the top m rows of $\widetilde{A}$ contribute a total value less than M to the dot product $\langle \widetilde{A}x, \widetilde{s} \rangle$ by (9), any solution to $\widetilde{A}, \widetilde{s}, \widetilde{t}$ must have support exactly n so that the resulting dot product contains the term nM.

It remains to prove correctness:

Claim B.1. *A vector* $y \in \{0,1\}^{2n}$ *satisfies* $\langle \widetilde{A}y, \widetilde{s} \rangle = \widetilde{t}$ *if and only if* $\mathrm{supp}(y) = n$ *and the first half of* y, *the vector* $y' = (y_1, y_2, \ldots, y_n)$, *satisfies* $\langle Ay', s \rangle = t$.

Proof: Suppose some vector $y' \in \{0,1\}^n$ satisfies $\langle Ay', s \rangle = t$. Then the vector y created by adding an arbitrary n-bit string with support $n - \mathrm{supp}(y')$ satisfies

$$\langle \widetilde{A}x', \widetilde{s} \rangle = \widetilde{t}$$

and is a valid solution to $\widetilde{A}, \widetilde{s}, \widetilde{t}$.

Now suppose some vector $y \in \{0,1\}^{2n}$ satisfies $\langle \widetilde{A}y, \widetilde{s} \rangle = \widetilde{t}$. As previously noted, we must have $\mathrm{supp}(y) = n$ to create the nM term in the product $\langle \widetilde{A}y, \widetilde{s} \rangle$. The additional Δ factors added to every component in each of the first m rows of $\widetilde{A}$ create the $n\Delta\|s\|_1$ term in the product. If we remove these two terms, the remainder of the equation $\langle \widetilde{A}y, \widetilde{s} \rangle = \widetilde{t}$ is $\langle A(y_1', y_2', \ldots, y_n'), s \rangle = t$. ∎

This concludes the proof of Observation B.2. □

B.3 Reduction of Binary ILP Feasibility to HBILP Feasibility

Proof of Lemma 5.4. Fix an instance $A \in \mathbb{Z}^{m \times n}, b \in \mathbb{Z}^n$ of Binary ILP Feasibility with $\Delta := \|A\|_\infty$. By Observation B.1, we can assume without loss of generality that the entries of A are non-negative.

Define $q := q(n, \Delta) = n\Delta + 1$ and create a new instance A, s, t' of HBILP feasibility by setting

$$s := (q^0, q^1, \ldots, q^{m-1}) \text{ and}$$
$$t' := \langle b, s \rangle = b_1 \cdot q^0 + b_2 \cdot q^1 + \cdots + b_m \cdot q^{m-1},$$

effectively using t' to store m registers of $\log_2(q) = \log_2(n\Delta + 1)$ bits each.

We claim $x \in \{0, 1\}^n$ solves A, b if and only if it solves A, s, t'. If $Ax = b$, $\langle Ax, s \rangle = t'$ follows immediately from the definition of t'.

Now suppose $\langle Ax, s \rangle = t'$. Because $q > A[i, \cdot]x$ for any row $i \in [m]$ by construction, the only way to achieve t' is if $A[i, \cdot]x = b_i$ for each $i \in [m]$. $\qquad \square$

C Proof of Lemma 7.1

The proof of Lemma 7.1 reformulates the well-known concepts of the *perfect hash family* and the *splitter*.

Definition C.1 (Splitter). An (n, k, ℓ)-splitter $\mathcal{F}$ is a family of functions from $[n]$ to $[\ell]$ such that for every set $S \subseteq [n]$ of size k, there exists $f \in \mathcal{F}$ such that for every $1 \leq j, j' \leq \ell$, the values $|f^{-1}(j) \cap S|$ and $|f^{-1}(j') \cap S|$ differ by at most 1.

In other words, for every $S \subseteq [n]$ of size k, some $f \in \mathcal{F}$ partitions $[n]$ into ℓ subsets in a way that splits S as evenly as possible. The special case of an (n, k, k)-splitter is called (n, k)-perfect hash family. We use the following construction of an (n, k)-perfect hash family due to Naor et al. [NSS95].

Theorem C.1 ([NSS95]). *For any $n, k \geq 1$, it is possible to construct an (n, k)-perfect hash family of size $e^k k^{O(\log k)} \log n$ in time $e^k k^{O(\log(k))} n \log n$.*

Observe that in our case, the $k^{O(\log k)}$ factor is absorbed by the $2^{O(k)}$ factor in the statement of Lemma 7.1. Let $A = \{a_1, \ldots, a_n\}$. For each function $f \in \mathcal{F}$ and each integer $i \in [k]$ we let $f_A : [k] \to 2^A$ be

$$f_A(i) = \{a_j \mid f(j) = i\}.$$

With the perfect hash family $\mathcal{F}$ we construct the set $\mathcal{P}$ as follows: for every function $f \in \mathcal{F}$ we simply add the set family $(f_A(1), \ldots, f_A(k))$ to the set $\mathcal{P}$. Observe that this set family forms a partition of A because f is a well-defined function. Theorem C.1 provides the claimed guarantees on the size of $\mathcal{P}$ and the construction time. Finally, let $S = \{x_1, \ldots, x_k\}$ be an arbitrary subset of A. Because $\ell = k$, Definition C.1 guarantees that for some $f \in \mathcal{F}$, $|f^{-1}(j) \cap S| = 1$ for every $j \in [k]$. This concludes the proof of Lemma 7.1.