Fine-Grained Equivalence for Problems Related to Integer Linear Programming

Lars Rohwedder
University of Southern Denmark (SDU), Odense, Denmark

Karol Węgrzycki
Saarland University, Saarbrücken, Germany
Max Planck Institute for Informatics, Saarbrücken, Germany

Abstract

Integer Linear Programming with n binary variables and m many 0/1-constraints can be solved in time $2^{\tilde{O}(m^2)}\text{poly}(n)$ and it is open whether the dependence on m is optimal. Several seemingly unrelated problems, which include variants of Closest String, Discrepancy Minimization, Set Cover, and Set Packing, can be modelled as Integer Linear Programming with 0/1 constraints to obtain algorithms with the same running time for a natural parameter m in each of the problems. Our main result establishes through fine-grained reductions that these problems are equivalent, meaning that a $2^{O(m^{2-\varepsilon})}\text{poly}(n)$ algorithm with $\varepsilon > 0$ for one of them implies such an algorithm for all of them.

In the setting above, one can alternatively obtain an $n^{O(m)}$ time algorithm for Integer Linear Programming using a straightforward dynamic programming approach, which can be more efficient if n is relatively small (e.g., subexponential in m). We show that this can be improved to ${n'}^{O(m)} + O(nm)$, where n' is the number of distinct (i.e., non-symmetric) variables. This dominates both of the aforementioned running times.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Integer Programming, Fine-Grained Complexity, Fixed-Parameter Tractable Algorithms

Digital Object Identifier 10.4230/LIPIcs.ITCS.2025.83

Related Version *Full Version*: https://arxiv.org/abs/2409.03675

Funding *Lars Rohwedder*: Supported by Dutch Research Council (NWO) project "The Twilight Zone of Efficiency: Optimality of Quasi-Polynomial Time Algorithms" [grant number OCEN.W.21.268]. *Karol Węgrzycki*: This work is part of the project TIPEA that has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (grant agreement No. 850979).

Acknowledgements We thank Friedrich Eisenbrand for his valuable insights and constructive discussions that enhanced this work.

1 Introduction

The study of parameterized complexity for Integer Linear Programming has a long history: classical works by Lenstra [15] and Kannan [12] and very recently Rothvoss and Reis [19] provide FPT algorithms in the number of variables n of an ILP of the form $Ax \leq b,\ x \in \mathbb{Z}^n$. In an orthogonal line of research, Papadimitriou [17] gave an FPT algorithm in the number of constraints and the size of the coefficients of A and b for an ILP in the standard form $Ax = b,\ x \in \mathbb{Z}^n_{\geq 0}$. Interest in the second line of work has been renewed by the improved algorithms due to Eisenbrand, Weismantel [9] and Jansen, Rohwedder [11], which give essentially optimal running times $(m\Delta)^{O(m)}\,\text{poly}(n)$ due to a conditional lower bound based on the Exponential Time Hypothesis (ETH) [14]. Here, Δ is the maximum absolute size

of an entry in A. The work by Eisenbrand and Weismantel also considers a version where variables are subject to box-constraints, which will be the primary focus of this work, see the definition below.

Integer Linear Programming
Input: Constraint matrix $A \in \{-\Delta, \dots, \Delta\}^{m \times n}$, right-hand side $b \in \mathbb{Z}^m$, variable bounds $\ell, u \in \mathbb{Z}^n_{\geq 0}$.
Task: Find $x \in \mathbb{Z}^n$ with

$$Ax = b$$
$$\ell_i \leq x_i \leq u_i \qquad i = 1, 2, \dots, n.$$

We refer to the variant where $\ell = (0, \dots, 0)^\mathsf{T}$ and $u = (1, \dots, 1)^\mathsf{T}$ as *binary* Integer Linear Programming.

Binary Integer Linear Programming
Input: Constraint matrix $A \in \{-\Delta, \dots, \Delta\}^{m \times n}$, right-hand side $b \in \mathbb{Z}^m$.
Task: Find $x \in \{0, 1\}^n$ with

$$Ax = b.$$

The running times obtained in [9] for either variant is $(m\Delta)^{O(m^2)} \operatorname{poly}(n)$. Also for matrices with only 0/1 coefficients nothing better than $2^{\tilde{O}(m^2)} \operatorname{poly}(n)$ is known. It is an intriguing question whether the slightly unusual exponent of $\tilde{O}(m^2)$ is necessary, which is in the spirit of *fine-grained complexity*. Since the dominant complexity-theoretic assumption of P≠NP is not powerful enough to show precise lower bounds on running times, the field of fine-grained complexity is concerned with finding lower bounds via stronger conjectures, see e.g., [2, 20, 6]. A number of such conjectures exist by now, often with interesting connections between them. Even if one doubts these conjectures, the reductions still provide insights into how problems relate to each other.

Based on existing conjectures, the best lower bound known on the exponent of Integer Linear Programming is $\Omega(m \log m)$ from the easier unbounded setting [14], which is of course not tight in this setting. In this paper, we take another path: we assume that Integer Linear Programming cannot be solved faster than the state-of-the-art and present several other natural parameterized problems that are all equivalent with respect to improvements on their running time.

▶ **Hypothesis 1** (ILP Hypothesis). *For every fixed $\varepsilon > 0$, there is no $2^{O(m^{2-\varepsilon})} \operatorname{poly}(n)$ time algorithm for Integer Linear Programming with $\Delta = O(1)$.*

In all of the problems below, the symbol m is chosen for the parameter of interest.

Closest String with Binary Alphabet
Input: Alphabet $\Sigma = \{0, 1\}$, strings $s_1, s_2, \dots, s_m \in \Sigma^n$
Task: Find string $t \in \Sigma^n$ minimizing

$$\max_i d(t, s_i),$$

where $d(t, s_i)$ is the Hamming distance between t and s_i, i.e., the number of positions the two strings differ in.

We refer to the generalization with arbitrary Σ simply as Closest String.

Discrepancy Minimization
Input: Universe $U = \{1, 2, \ldots, n\}$, set system $S_1, S_2, \ldots, S_m \subseteq U$
Task: Find coloring $\chi : U \to \{-1, 1\}$ minimizing

$$\max_i \left| \sum_{u \in S_i} \chi(u) \right|.$$

Set Multi-Cover
Input: Universe $U = \{1, 2, \ldots, m\}$, set system $S_1, S_2, \ldots, S_n \subseteq U$, $b \in \mathbb{N}$
Task: Find $I \subseteq \{1, 2, \ldots, n\}$ of minimal cardinality such that for each $v \in U$ there are at least b sets S_i with $i \in I$ and $v \in S_i$.

Set Multi-Packing
Input: Universe $U = \{1, 2, \ldots, m\}$, set system $S_1, S_2, \ldots, S_n \subseteq U$, $b \in \mathbb{N}$
Task: Find $I \subseteq \{1, 2, \ldots, n\}$ of maximal cardinality such that for each $v \in U$ there are at most b sets S_i with $i \in I$ and $v \in S_i$.

Many of these problems are well-known applications of ILP techniques. For example, Knop et al. [13] provided $m^{O(m^2)} \cdot \text{poly}(n)$ algorithms for Closest String and Set Multi-Cover. Discrepancy Minimization is usually considered within the context of approximation algorithms (see, e.g., [3]).

As mentioned above, our main result is the following equivalence.

▶ **Theorem 2.** *The following statements are equivalent:*
(1) *There exists an* $2^{O(m^{2-\varepsilon})} \text{poly}(n)$ *algorithm for Integer Linear Programming with* $\Delta = O(1)$ *with* $\varepsilon > 0$.
(2) *There exists an* $2^{O(m^{2-\varepsilon})} \text{poly}(n)$ *algorithm for Binary Integer Linear Programming with* $A \in \{0, 1\}^{m \times n}$ *and* $n \leq m^{O(m)}$ *with* $\varepsilon > 0$.
(3) *There exists an* $2^{O(m^{2-\varepsilon})} \text{poly}(n)$ *algorithm for Closest String with Binary Alphabet with* $\varepsilon > 0$.
(4) *There exists an* $2^{O(m^{2-\varepsilon})} \text{poly}(n)$ *algorithm for Discrepancy Minimization with* $\varepsilon > 0$.
(5) *There exists an* $2^{O(m^{2-\varepsilon})} \text{poly}(n)$ *algorithm for Set Multi-Cover with* $\varepsilon > 0$.
(6) *There exists an* $2^{O(m^{2-\varepsilon})} \text{poly}(n)$ *algorithm for Set Multi-Packing with* $\varepsilon > 0$.

Note that Item 1 is the negation of Hypothesis 1. All problems in Theorem 2 are easily transformed into the first problem, i.e., Integer Linear Programming with $\Delta = O(1)$, while maintaining the same value of m. Hence, the more interesting aspect of the theorem is that all these problems are as expressive as the first one.

Hypothesis 1 considers Integer Linear Programming with relatively small entries, i.e., $\Delta = O(1)$. One can also ask the question of whether there is any parameter regime for Δ for which the state-of-the-art can be improved. In this spirit, a stronger variant of the conjecture is the following.

▶ **Hypothesis 3** (Strong ILP Hypothesis). *For every fixed* $\varepsilon > 0$ *and* $\delta \geq 0$, *there is no* $2^{O(m^{\delta+2-\varepsilon})} \text{poly}(n)$ *time algorithm for Integer Linear Programming with* $\Delta = 2^{m^\delta}$.

Note that Hypothesis 1 is a special case of Hypothesis 3 for $\delta = 0$. Another interesting regime is the complexity of Integer Linear Programming with $\Delta = 2^m$, because of a connection to block-structure integer programming, which we elaborate on later. There, the state-of-the-art algorithm requires time $m^{O(m^3)} \text{poly}(n)$, Integer Linear Programming with large entries can be reduced to an equivalent instance with a 0/1 matrix as seen in the following theorem, but the reduction is not strong enough to show equivalence between the two hypotheses.

▶ **Theorem 4.** *There is a polynomial time algorithm that transforms an instance of* Integer Linear Programming *with* $\Delta > 1$ *into an equivalent one with* $A' \in \{0,1\}^{m' \times n'}$ *for* $m' = O(m \log \Delta)$ *and* $n' \leq {m'}^{O(m')}$.

This implies that if there is an algorithm with running time $2^{O(m^{1.5-\varepsilon})} \operatorname{poly}(n)$ for Integer Linear Programming with $A \in \{0,1\}^{m \times n}$, then there is a $2^{O(m^{3-\varepsilon'})} \operatorname{poly}(n)$ time algorithm for Integer Linear Programming with $\Delta = 2^m$.

One might hope to improve the theorem to $m' = O(m\sqrt{\log \Delta})$, since then a $2^{O(m^{2-\varepsilon})} \operatorname{poly}(n)$ time algorithm for 0/1 matrices would imply a $2^{O(m^{3-\varepsilon'})} \operatorname{poly}(n)$ time algorithm for $\Delta = 2^m$. However, such a reduction would imply the strong result that under ETH is equivalent to Hypothesis 1. This is because under ETH the Subset Sum problem, i.e., the case when $m = 1$, cannot be solved in $\Delta^{o(1)} \operatorname{poly}(n)$ time [1] and the hypothetical reduction would be able to encode an instance of Subset Sum into an ILP with $m = O(\sqrt{\log \Delta})$. We are not aware of any meaningful reduction in the other direction, i.e., from large m and small Δ to smaller m and larger Δ. It is possible to aggregate m constraints into a single one with entries bounded by $\Delta' = \Delta^{O(m^2)}$, but this reduction seems useless since the resulting parameter range requires $\operatorname{poly}(\Delta') \cdot \operatorname{poly}(n)$ time due to the ETH lower bound mentioned above.

Assuming Hypothesis 3, we derive a tight lower for a form of block-structured integer linear programs that has been studied extensively in recent literature. For simplicity, we consider here the basic setting with $m \times m$ submatrices.

n-Fold Integer Linear Programming
Input: Square matrices $A_1, \ldots, A_n, B_1, \ldots, B_n \in \{-\Delta, \ldots, \Delta\}^{m \times m}$, right-hand sides $b^{(0)}, \ldots, b^{(n)} \in \mathbb{Z}^m$.
Task: Find $x^{(1)}, \ldots, x^{(n)} \in \mathbb{Z}^m_{\geq 0}$ with

$$\begin{aligned} A_1 x^{(1)} + \ldots + A_n x^{(n)} &= b^{(0)} \\ B_1 x^{(1)} &= b^{(1)} \\ &\vdots \\ B_n x^{(n)} &= b^{(n)} \end{aligned}$$

▶ **Theorem 5.** *For every* $\delta > 0$*, there is no algorithm with running time* $2^{O(m^{3-\delta})} \operatorname{poly}(n)$ *for* n*-Fold Integer Linear Programming when the maximum absolute entry is bounded by* $\Delta = O(1)$*, unless Hypothesis 3 is false.*

This matches the best algorithms known for the problem, see [5] and references therein. The reduction follows the same idea as used in [10], where the authors show a non-tight quadratic lower bound for the exponent based on ETH. Our lower bound is stronger simply because the conjecture we base it on is stronger.

1.1 Tightness of more general problems

There are several other, more general problems to the ones mentioned above, for which known algorithms would be tight assuming that one cannot improve the running time for the class of problems in Theorem 2. However, we do not know if these are all equivalent.

The algorithm by Eisenbrand and Weismantel [9] also works for the optimization version of ILP, i.e.,

$$\max c^\mathsf{T} x,\ Ax = b,\ \ell \le x \le u,\ x \in \mathbb{Z}^n.$$

This leads to the same running time of $(m\Delta)^{O(m^2)}\operatorname{poly}(n)$ except for a slightly higher constant in the exponent. Notably, the coefficients of c do not increase the number of arithmetic operations.

Given a matrix $A \in \{-\Delta, \dots, \Delta\}^{m\times n}$ and a convex function $g : \mathbb{R}^m \to \mathbb{R} \cup \{\infty\}$, Dadush, Léonard, Rohwedder, and Verschae [7] have shown that one can find the minimizer of $g(Ax),\ x \in \{0,1\}^n$ in time $(m\Delta)^{O(m^2)}\operatorname{poly}(n)$ (assuming polynomial time to evaluate g). This problem is a generalization of Binary Integer Linear Programming, since one can take $g(b') = 0$ if $b' = b$ and $g(b') = \infty$ otherwise.

Given a linear matroid $M = (E, \mathcal{I})$, where $|E| = n$, a matrix $A \in \{-\Delta, \dots, \Delta\}^{n\times m}$ and a right-hand side $b \in \mathbb{Z}^m$, Eisenbrand, Rohwedder, and Węgrzycki [8] gave an algorithm that finds a basis B of M such that $Ax_B = b$ in time $O(m\Delta)^{O(m^2)}\operatorname{poly}(n)$, where x_B is the incidence vector of B. This generalizes Binary Integer Linear Programming, since one can take E as the set of binary variables and n additional dummy variables and then take M as the uniform matroid of rank n.

Finally, Closest String (with arbitrary alphabet) can still be solved in time $m^{O(m^2)}\operatorname{poly}(n)$ for example by casting it as the previous matroid problem [8] or as a block structured ILP, see [13].

1.2 Algorithm for few distinct variables

We show that the running time of $2^{\tilde{O}(m^2)}\operatorname{poly}(n)$ for Integer Linear Programming with $\Delta = O(1)$, i.e., the subject of Hypothesis 1, can be improved if the number of distinct variables is low. For the sake of generality, we state the algorithmic result for the optimization variant and any range of Δ.

▶ **Theorem 6.** *Consider the integer programming problem*

$$\begin{aligned} &\max\ c^\mathsf{T} x \\ &\quad Ax = b \\ &\ell_i \le x_i \le u_i && i = 1, \dots, n \\ &\quad x \in \mathbb{Z}^n\ . \end{aligned} \tag{1}$$

Let Δ be an upper bound on the absolute value of entries of A. The problem (1) can be solved in time

$$n^{m+1} \cdot O(m\Delta)^m \cdot \log(\|u - \ell\|_\infty).$$

Using standard reductions, see e.g. Section 2.5.2, one may reduce $\|u - \ell\|_\infty$ to $(m\Delta)^{O(m)}$, making the logarithmic factor insignificant.

We will now interpret this running time and point out interesting special cases.

Binary ILP with $A \in \{0,1\}^{m\times n}$

Here, the running time above implies an $n'^{O(m)} + O(nm)$ time algorithm, where n' is the number of distinct variables, i.e., variables that differ either in their entry of c or in their column of A: one can merge identical variables in time $2^{O(m)} + O(nm)$ time, after which

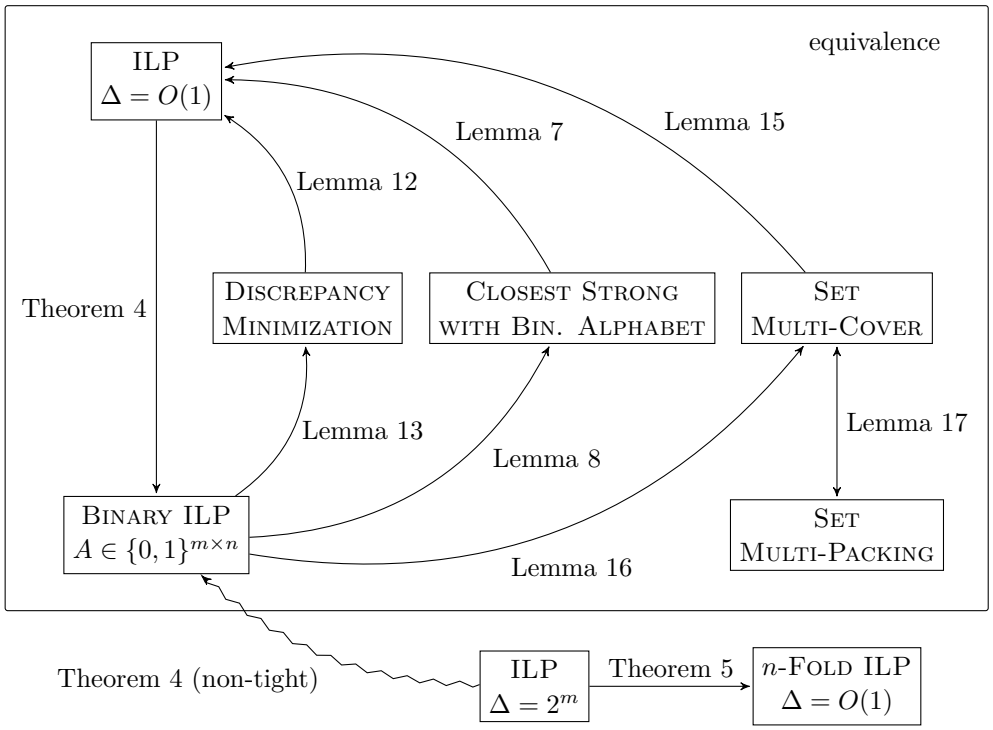

Figure 1 Overview of reductions in this paper.

$\|u\|_\infty \leq n$. Furthermore, without loss of generality, the rows of A are linearly independent, thus $m \leq n'$. Hence, the overall running time is

$$\begin{aligned} 2^{O(m)} + O(nm) + {n'}^{m+1} \cdot O(m)^m \cdot \log n &\leq {n'}^{O(m)} \cdot \log n + O(nm) \\ &\leq {n'}^{O(m)} + O(nm). \end{aligned}$$

Here, the last inequality follows from a case distinction whether $\log n$ is greater than ${n'}^{O(m)}$ or not. Note that in the setting without objective function we have $n' \leq 2^m$. Thus, this running time is at least as good as the known $2^{\tilde{O}(m^2)} \operatorname{poly}(n)$ time algorithms. It also dominates the running time of $n^{O(m)}$ one would get by simple dynamic programming over the right-hand sides b' that are feasible (for which there can be at most n^m).

Binary ILP with $A \in \{0,1\}^{m \times n}$ and a constant number of 1s in each column

Here, the number of distinct columns is polynomial in m and the previous case implies a running time of $m^{O(m)} + O(nm)$, meaning that Hypothesis 1 does not extend to this special case.

2 Reductions

The basic idea of all reductions for Theorem 2 is that we transform one problem with parameter m into another problem with parameter $m' = O(m \cdot \log^{O(1)} m)$. Additionally the size of the instance may only increase from n to $2^{O(m^{2-\delta})} \operatorname{poly}(n)$ for some $\delta > 0$. The concrete reductions we prove can be seen in Figure 1.

2.1 Closest String

Let d be the bound on the maximum hamming distance in the decision version of Closest String with Binary Alphabet. For a string $s \in \Sigma^n$ we denote by $s[i]$ the ith character. For $i \leq j$ we write $s[i \dots j]$ for the substring from the ith to the jth character.

▶ **Lemma 7.** *Theorem 2, Statement 1 implies Statement 3*

Proof. The following is an ILP model for Closest String with Binary Alphabet.

$$\sum_{i=1}^{n} x_i \cdot \mathbb{1}_{\{s_j[i]=0\}} + (1-x_i) \cdot \mathbb{1}_{\{s_j[i]=1\}} \leq d \qquad \text{for all } j \in \{1,2,\dots,m\}$$

$$x \in \{0,1\}^n$$

One may add slack variables to turn the inequalities into equalities. ◀

▶ **Lemma 8.** *Theorem 2, Statement 3 implies Statement 2*

Proof. We want to transform the following ILP

$$\begin{aligned} Ax &= b \\ x &\in \{0,1\}^n \end{aligned} \tag{2}$$

where $A \in \{0,1\}^{m \times n}$, into an equivalent instance of Closest String. We first rewrite (2) into a more convenient form.

▷ Claim 9. One can in polynomial time construct a matrix $C \in \{-1,1\}^{(2m+2)\times 2n}$ and some $c \in \mathbb{Z}^{2m+2}$ such that (2) is feasible if and only if there is a solution to

$$\begin{aligned} Cx &\leq c \\ x &\in \{0,1\}^{2n}. \end{aligned} \tag{3}$$

Furthermore, every feasible x for (3) has $x_1 + \dots + x_{2n} = n$.

This follows from simple transformations. We defer the proof until later and first show how the reduction follows from it. By comparing to the ILP formulation of Closest String we observe that (3) corresponds to a "non-uniform" variant of Closest String. It can be reformulated as: given strings $s_1, \dots, s_{2m+2} \in \{0,1\}^{2n}$ and bounds $d_1, \dots, d_{2m+2} \in \mathbb{Z}$, find a string $t \in \{0,1\}^{2n}$ such that for each $j = 1, \dots, m$ we have $d(t, s_j) \leq d_j$. This follows from the ILP model given in the proof of Lemma 7. Furthermore, we have the guarantee that any solution has exactly n ones. To transform this into a regular instance, we add two more strings r_1, r_2 and to each string s_j we add $4n$ more characters, which makes a total of $6n$ characters per string. The strings of this instance $s'_1, \dots, s'_{2m+2}, r_1, r_2$ are defined as

$$\begin{aligned} r_1 &= (\ 0,\dots,0,\ 0,\dots,0,\ 0,\dots,0,\ 0,\dots,\dots,0\ ), \\ r_2 &= (\ 1,\dots,1,\ 1,\dots,1,\ 1,\dots,1,\ 0,\dots,\dots,0\ ), \\ s'_j &= (\ \underbrace{s_j}_{2n \text{ chars}},\ \underbrace{1,\dots,1}_{n \text{ chars}},\ \underbrace{0,\dots,0}_{n \text{ chars}},\ \underbrace{1,\dots,1}_{2n-d_j \text{ chars}}\ \underbrace{0,\dots,0}_{d_j \text{ chars}}\ ). \end{aligned}$$

Here, we assume without loss of generality that $d_j \in \{0,\dots,2n\}$. We claim that there is a solution to this instance with maximum Hamming distance $2n$ if and only if there is a solution to the non-uniform instance.

▷ Claim 10. If there is a string $t' \in \{0,1\}^{6n}$ with distance at most $2n$ to r_1, r_2, and s'_j, $j = 1, 2, \ldots, 2m+2$, then there is also a string $t \in \{0,1\}^{2n}$ with distance at most d_j to s_j, $j = 1, 2, \ldots, 2m+2$.

▷ Claim 11. If there is a string $t \in \{0,1\}^{2n}$ with distance at most d_j to s_j, $j = 1, 2, \ldots, 2m+2$, then there is also a string $t' \in \{0,1\}^{6n}$ with distance at most $2n$ to r_1, r_2, and s'_j, $j = 1, 2, \ldots, 2m+2$.

From these claims, the lemma follows immediately. ◀

Proof of Claim 9. We add variables $\bar{x}_1, \ldots, \bar{x}_n$ and force a solution to take exactly n many ones

$$\begin{aligned} Ax &= b \\ x_1 + \cdots + x_n + \bar{x}_1 + \cdots + \bar{x}_n &= n \\ x, \bar{x} &\in \{0,1\}^n. \end{aligned}$$

Next, we change the equality constraints into inequalities and A into a $-1, 1$ matrix

$$\begin{aligned} A'x - \mathbf{1}\bar{x} &\leq b' \\ -A'x + \mathbf{1}\bar{x} &\leq -b' \\ x_1 + \cdots + x_n + \bar{x}_1 + \cdots + \bar{x}_n &\leq n \\ -x_1 - \cdots - x_n - \bar{x}_1 - \cdots - \bar{x}_n &\leq -n \\ x, \bar{x} &\in \{0,1\}^n \end{aligned}$$

where $A' = 2A - \mathbf{1}$ with $\mathbf{1}$ being the all-ones $m \times n$ matrix and $b' = 2b - (n, \ldots, n)^\mathsf{T}$. ◁

Proof of Claim 10. Suppose there is a string $t' \in \{0,1\}^{6n}$ with distance at most $2n$ to each of the strings $s'_1, \ldots, s'_{2m+2}, r_1, r_2$. Because $d(r_1, r_2) = 4n$, string t' must match r_1 and r_2 on characters where r_1 and r_2 match. More precisely, $t'[i] = 0$ for $i \in \{4n, \ldots, 6n\}$. Formally, this follows because of

$$\begin{aligned} 4n &\geq d(t', r_1) + d(t', r_2) \\ &= d(t'[1 \ldots 4n], r_1[1 \ldots 4n]) + d(t'[1 \ldots 4n], r_2[1 \ldots 4n]) \\ &\quad + 2d(t'[4n+1 \ldots 6n], (0, \ldots, 0)) \\ &\geq d(r_1[1 \ldots 4n], r_2[1 \ldots 4n]) + 2d(t'[4n+1 \ldots 6n], (0, \ldots, 0)) \\ &\geq 4n + 2d(t'[4n+1 \ldots 6n], (0, \ldots, 0)). \end{aligned}$$

Let us now analyze the distance between $t := t'[1 \ldots 2n]$ and s_j for some j. Since the last $2n$ characters of t' are zero, we have $d(t'[4n+1 \ldots 6n], s'_j[4n+1 \ldots 6n]) = 2n - d_j$. Thus,

$$\begin{aligned} d(t, s_j) &\leq d(t', s'_j) - d(t'[4n+1 \ldots 6n], s'_j[4n+1 \ldots 6n]) \\ &\leq 2n - (2n - d_j) = d_j. \end{aligned}$$

Thus, string t is a solution for the non-uniform instance. ◁

Proof of Claim 11. Let $t \in \{0,1\}^{2n}$ with $d(t[1 \ldots 2n], s_j) \leq d_j$ for all j. We extend it to a string $t' \in \{0,1\}^{6n}$ by setting $t'[1 \ldots 2n] = t$, $t[2n+1 \ldots 3n] = (1, \ldots, 1)$, and $t[3n+1 \ldots 6n] = (0, \ldots, 0)$. Let us now verify that t' has a distance at most $2n$ to each string. For r_1, r_2 note that $t[1 \ldots 2n]$ has exactly n ones by guarantee of the non-uniform instance. Thus, the distance to r_1 and r_2 is exactly $2n$. Consider some $j \in \{1, \ldots, 2m+2\}$. Then

$$d(s'_j, t) = d(s_j, t[1 \ldots 2n]) + 2n - d_j \leq 2n.$$ ◀

2.2 Discrepancy Minimization

▶ **Lemma 12.** *Theorem 2, Statement 1 implies Statement 4*

Proof. Let d be a bound on the objective in the decision version of Discrepancy Minimization. Let A be the incidence matrix of the given set system. Then the colorings of discrepancy at most d are exactly the solutions of

$$\begin{pmatrix} A \\ -A \end{pmatrix} y \leq \begin{pmatrix} d \\ d \\ \vdots \\ d \end{pmatrix}$$
$$y \in \{-1,1\}^n.$$

This can be equivalently formulated as

$$\begin{pmatrix} 2A \\ -2A \end{pmatrix} x \leq \begin{pmatrix} A \\ -A \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} + \begin{pmatrix} 2d \\ 2d \\ \vdots \\ 2d \end{pmatrix}$$
$$x \in \{0,1\}^n.$$

where $x_i = (1+y_i)/2$. One may translate the inequalities into equalities by introducing slack variables. Therefore, an algorithm for Integer Linear Programming can be used to solve Discrepancy Minimization. ◀

▶ **Lemma 13.** *Theorem 2, Statement 4 implies Statement 2.*

Proof. Consider an ILP of the form

$$Ax = b \tag{4}$$
$$x \in \{0,1\}^n$$

for $A \in \{0,1\}^{m \times n}$ and $n \leq m^{O(m)}$. We will construct an instance of Discrepancy Minimization which has discrepancy zero if and only if the ILP has a feasible solution. Towards this, we first reformulate the ILP above as

$$Ay = b' \tag{5}$$
$$y \in \{-1,1\}^n$$

where $b' = 2b - A\,(1,\dots,1)^\mathsf{T}$. Note that x is feasible for (4) if and only if $y = 2x - (1,\dots,1)^\mathsf{T}$ is feasible for (5). Also, if $b' = (0,\dots,0)^\mathsf{T}$, then (5) is already equivalent to an instance of Discrepancy Minimization that tests for discrepancy zero. To handle the general case, we transform it into an equivalent system with right-hand size $(0,\dots,0)^\mathsf{T}$. We first construct a gadget of elements that have the same color.

▷ Claim 14. For any $k \in \mathbb{N}$ we can construct a pair of matrices $B, \bar{B} \in \{0,1\}^{(2k-1)\times 2^k}$ such that there are exactly two solutions to

$$Bz + \bar{B}\bar{z} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$$
$$z, \bar{z} \in \{-1,1\}^{2^k}$$

namely $z = (1,\dots,1)^\mathsf{T}, \bar{z} = (-1,\dots,-1)^\mathsf{T}$ and $z = (-1,\dots,-1)^\mathsf{T}, \bar{z} = (1,\dots,1)^\mathsf{T}$.

We will defer the proof to the end of the section. Using this gadget with $k = \lceil \log_2 n \rceil = O(m \log m)$ we now replace each coefficient b'_j in the previous system by the variables from the gadget. Note that (5) is infeasible if $\|b'\|_\infty > n$. Thus assume without loss of generality that $\|b'\|_\infty \leq n \leq 2^k$. Let $C, \bar{C} \in \{0,1\}^{2^k \times m}$ be defined as follows. The jth row of C has b'_j many ones at arbitrary positions if $b'_j \geq 0$ and is all zero otherwise; contrary, the jth row of $\bar{C}$ has $-b'_j$ many ones at arbitrary positions if $b'_j < 0$ and is all zero otherwise.

Now consider the system

$$\begin{aligned} Ay + Cz + \bar{C}\bar{z} &= \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \\ Bz + \bar{B}\bar{z} &= \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \\ y &\in \{-1,1\}^n \\ z, \bar{z} &\in \{-1,1\}^{2^k}. \end{aligned} \tag{6}$$

We claim that (6) has a solution if and only if there is a solution to (5). Let $y, z, \bar{z}$ be a solution to the former. Notice that the negation of a solution is also feasible. Due to Claim 14 we may assume without loss of generality that $z = (-1, \ldots, -1)^\mathsf{T}$ and $\bar{z} = (1, \ldots, 1)^\mathsf{T}$, negating the solution if necessary. It follows that

$$Cz + \bar{C}\bar{z} = -b'.$$

Thus, $Ay = b'$, which concludes the first direction. For the other direction, assume that there is a solution y to (5). We set $z = (-1, \ldots, -1)^\mathsf{T}$, $\bar{z} = (1, \ldots, 1)^\mathsf{T}$, which by Claim 14 satisfies $Bz + \bar{B}\bar{z} = (0, \ldots, 0)^\mathsf{T}$. As before we have that $Cz + \bar{C}\bar{z} = -b'$. Thus, $y, z, \bar{z}$ is a solution to (6). This establishes the equivalence of the initial ILP instance to (6), which corresponds to an instance of Discrepancy Minimization where we test for discrepancy zero with $m' = O(m \log m)$ sets. ◀

Proof of Claim 14. The existence of such a matrix can be proven by induction: for $k = 1$, we simply take $B = \bar{B} = (1)$. Now suppose that we already have a pair of matrices $B, \bar{B} \in \{0,1\}^{(2k-1)\times 2^k}$ as above. Then we set

$$B' = \begin{pmatrix} B & 0 \\ 1\cdots 1 & 0\cdots 0 \\ 0\cdots 0 & 1\cdots 1 \end{pmatrix} \text{ and } \bar{B}' = \begin{pmatrix} \bar{B} & 0 \\ 0\cdots 0 & 1\cdots 1 \\ 1\cdots 1 & 0\cdots 0 \end{pmatrix} \in \{0,1\}^{(2k+1)\times 2\cdot 2^k}.$$

It can easily be checked that choosing either $z = z' = (1, \ldots, 1)^\mathsf{T}, \bar{z} = \bar{z}' = (-1, \ldots, -1)^\mathsf{T}$ or $z = z' = (-1, \ldots, -1)^\mathsf{T}, \bar{z} = \bar{z}' = (1, \ldots, 1)^\mathsf{T}$ satisfies

$$B' \begin{pmatrix} z \\ z' \end{pmatrix} + \bar{B}' \begin{pmatrix} \bar{z} \\ \bar{z}' \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{7}$$

Now take any $z, z', \bar{z}, \bar{z}' \in \{-1,1\}^{2^{k+1}}$ that satisfy (7). Then we have that $Bz + \bar{B}\bar{z} = (0, \ldots, 0)^\mathsf{T}$. Hence by induction hypothesis either $z = (1, \ldots, 1)^\mathsf{T}, \bar{z} = (-1, \ldots, -1)^\mathsf{T}$ or $z = (-1, \ldots, -1)^\mathsf{T}, \bar{z} = (1, \ldots, 1)^\mathsf{T}$. Assume for now the first case holds. Then because of the second-to-last rows of $B, \bar{B}$ we have

$$2^k + \sum_{i=1}^{2^k} \bar{z}'_i = \sum_{i=1}^{2^k} z_i + \bar{z}'_i = 0.$$

Hence, $\bar{z}' = (-1, \dots, -1)^\mathsf{T} = \bar{z}$. Similarly, the last row of $B, \bar{B}$ implies that $z' = (1, \dots, 1)^\mathsf{T} = z$. Analogously, if $z = (-1, \dots, -1)^\mathsf{T}, \bar{z} = (1, \dots, 1)^\mathsf{T}$ then $z' = z$ and $\bar{z}' = \bar{z}$. ◁

2.3 Set Multi-Cover

▶ **Lemma 15.** *Theorem 2, Statement 1 implies Statement 5.*

Proof. An instance of the decision version of Set Multi-Cover with a bound d on the cardinality can be formulated as

$$\begin{aligned} x_1 + \cdots + x_n &\leq d \\ Ax &\geq \begin{pmatrix} b \\ \vdots \\ b \end{pmatrix} \\ x &\in \{0,1\}^n \end{aligned}$$

where $A \in \{0,1\}^{m \times n}$ is the incidence matrix of the given set system. This can easily be translated to the form of (1) by introducing slack variables. Notice that this ILP has $m + 1$ constraints. Thus, a faster algorithm for ILP would imply a faster algorithm for Set Multi-Cover. ◀

In the remainder, we show that the converse is also true.

▶ **Lemma 16.** *Theorem 2, Statement 5 implies Statement 2.*

Proof. First, we reduce to a "non-uniform" version of Set Multi-Cover where each element v has a different demand b_v. Let $A \in \{0,1\}^{m \times n}$ and $b \in \mathbb{Z}^m_{\geq 0}$ and consider the solutions to

$$\begin{aligned} Ax &= b \\ x &\in \{0,1\}^n. \end{aligned}$$

First, we add n additional binary variables $\bar{x}$ and the requirement that exactly n variables are equal to one, i.e.,

$$\begin{aligned} Ax &= b \\ x_1 + \cdots + x_n + \bar{x}_1 + \cdots + \bar{x}_n &= n \\ x &\in \{0,1\}^n \\ \bar{x} &\in \{0,1\}^n. \end{aligned}$$

This system is equivalent to the previous one, by setting $n - x_1 - \cdots - x_n$ arbitrary variables $\bar{x}_i$ to 1. Next, we transform the equality constraints $Ax = b$ into inequalities:

$$
\begin{aligned}
Ax &\geq b \\
(\mathbf{1} - A)x + \mathbf{1}\bar{x} &\geq \begin{pmatrix} n \\ \vdots \\ n \end{pmatrix} - b \\
x_1 + \cdots + x_n + \bar{x}_1 + \cdots + \bar{x}_n &= n \\
x &\in \{0,1\}^n \\
\bar{x} &\in \{0,1\}^n.
\end{aligned}
$$

Here, $\mathbf{1}$ denotes the $n \times n$ all-ones matrix. Note that the second constraint is equivalent to $Ax \leq b$, since we fixed the number of ones in the solution. This ILP is feasible if and only if the optimum of the following ILP is n.

$$
\begin{aligned}
\min\ & x_1 + \cdots + x_n + \bar{x}_1 + \cdots + \bar{x}_n \\
& Ax \geq b \\
& (\mathbf{1} - A)x + \mathbf{1}\bar{x} \geq \begin{pmatrix} n \\ \vdots \\ n \end{pmatrix} - b \\
& x_1 + \cdots + x_n + \bar{x}_1 + \cdots + \bar{x}_n \geq n \\
& x \in \{0,1\}^n \\
& \bar{x} \in \{0,1\}^n.
\end{aligned}
$$

This ILP corresponds to an instance of non-uniform Set Multi-Cover with $2m+1$ elements.

To reduce a non-uniform instance $S_1, \ldots, S_n$, $(b_v)_{v \in U}$, to a uniform instance of Set Multi-Cover we proceed as follows: add one new element and n many new sets to the instance. The coverage requirement of each element is n. The new element is contained in each of the new sets and in none of the old ones. Thus, each new set has to be taken. Furthermore, we add each old element v to $n - b_v$ many arbitrary new sets. ◀

2.4 Set Multi-Packing

▶ **Lemma 17.** *Theorem 2, Statements 5 and 6 are equivalent.*

Proof. Notice the following duality between Set Multi-Cover and Set Multi-Packing. Let $U = \{1, 2, \ldots, m\}$, $S_1, S_2, \ldots, S_n$, and $b \in \mathbb{N}$ be an instance of Set Multi-Cover. Now consider the instance of Set Multi-Packing with universe U, set system $\bar{S}_1 = U \setminus S_1, \bar{S}_2 = U \setminus S_2, \ldots, \bar{S}_n = U \setminus S_n$, and bounds $\bar{b} = n - b$. This is a bipartition between instances of Set Multi-Cover and Set Multi-Packing, i.e., it can be performed in both ways.

For one pair of such instances, a solution I for Set Multi-Cover is feasible if and only if $\bar{I} = \{1, 2, \ldots, n\} \setminus I$ is feasible for Set Multi-Packing. Thus, if the optimum of Set Multi-Cover is k, then the optimum of Set Multi-Packing is $n - k$. ◀

2.5 Integer Linear Programming

In this section, we prove Theorem 4, i.e., we show how to reduce an ILP with large coefficients into a (larger) ILP with only 0/1 coefficients.

▶ **Theorem 4.** *There is a polynomial time algorithm that transforms an instance of* Integer Linear Programming *with* $\Delta > 1$ *into an equivalent one with* $A' \in \{0,1\}^{m' \times n'}$ *for* $m' = O(m \log \Delta)$ *and* $n' \leq {m'}^{O(m')}$.

Furthermore, we show how to reduce an ILP with arbitrary upper and lower bounds into one with at most $(m\Delta)^{O(m)}$ binary variables. Note that this implies that Statements 1 and 2 are equivalent and concludes the proof of Theorem 2.

2.5.1 From large coefficients to zero-one

We will transform an ILP of the form

$$\begin{aligned} Ax &= b \\ \ell \leq x &\leq u \\ x &\in \mathbb{Z}^n \end{aligned}$$

where $A \in \{-\Delta, \dots, \Delta\}^{m \times n}$ into an equivalent one with $A' \in \{0,1\}^{m' \times n'}$ for $m' = O(m \log \Delta)$. Let $k = \lceil \log_2(\Delta) \rceil$. For the jth row of A we introduce $4(k+1)$ rows in A', denoted by $r_0^+(j), {r'}_0^+(j), \dots, r_k^+(j), {r'}_k^+(j), r_0^-(j), {r'}_0^-(j), \dots, r_k^-(j), {r'}_k^-(j)$. Intuitively, the rows $r_i^+(j), {r'}_i^+(j)$ are symmetric rows that each stand for a value of 2^i in row j. Similarly, $r_i^-(j), {r'}_i^-(j)$ stand for a value of -2^i in row j. The right-hand sides of the new ILP are b_j for row $r_0^+(j)$ and zero for all other rows affiliated with j.

For each column A_i we derive a column of A' as follows: for row j we consider the binary encoding of A_{ji} and have the column in A' use this binary encoding in $r_0^+(j), r_1^+(j), \dots, r_k^+(j)$ if $A_{ji} \geq 0$ and in $r_0^-(j), r_1^-(j), \dots, r_k^-(j)$ if $A_{ji} < 0$. All other entries of this column are zero.

We now add auxiliary variables to shift the values from one power to another. For each row j and each $i = 0, \dots, k-1$ we add:

- a variable $x(i,j)$ with a one in row $r_i^+(j), {r'}_i^+(j)$ and $r_{i+1}^-(j)$ and
- a variable $\bar{x}(i,j)$ with a one in row $r_i^-(j), {r'}_i^-(j)$ and $r_{i+1}^+(j)$.

Furthermore, for each row j and each $i = 0, \dots, k$ we add:

- a variable $x_a(i,j)$ with a one in row ${r'}_i^+(j)$ and $r_i^-(j)$,
- a variable $x_b(i,j)$ with a one in row ${r'}_i^-(j)$ and $r_i^+(j)$, and
- a variable $x_c(i,j)$ with a one in row $r_i^-(j)$ and $r_i^+(j)$.

Note that each auxiliary variable does not alter the total value for row j, i.e., the sum of 2^i times $r_i^+(j)$ and ${r'}_i^+(j)$ minus 2^i times $r_i^-(j)$ and ${r'}_i^-(j)$ across all i.

Thus, any solution to the new ILP must correspond to a solution of the original ILP. The converse is also true: the auxiliary variables can be adjusted to preserve the original value of the ILP. This is because any value for one of the rows of j can always be shifted to $r_0^+(j)$ using the auxiliary variables, while still satisfying the constraints. Therefore, we can enforce the value of the unchanged variables.

2.5.2 From bounded to binary variables

Consider an ILP of the form

$$\begin{aligned} Ax &= b \\ \ell \leq x &\leq u \\ x &\in \mathbb{Z}^n \end{aligned}$$

where $A \in \{-\Delta, \dots, \Delta\}^{m \times n}$. We will first transform this into an equivalent ILP of the form

$$\begin{aligned} A'x &= b' \\ x &\in \{0,1\}^{n'} \end{aligned} \tag{8}$$

where $A' \in \{-\Delta, \dots, \Delta\}^{m \times n'}$ for $n' \leq (m\Delta)^{O(m)}$. Assume without loss of generality that each column of A is different. Otherwise, we can merge identical columns together by adding the respective upper and lower bounds. This implies that $n \leq (2\Delta+1)^m$. Next, we compute a vertex solution x^* to the continuous relaxation $\{Ax = b, \ell \leq x \leq u, x \in \mathbb{R}^n\}$. The proximity bound by Eisenbrand and Weismantel [9] shows that there is an integer solution z with $\|z - x^*\|_1 \leq m(2m\Delta+1)^m$ if any integer solution exists. Thus, choosing $\ell'_i = \max\{\ell_i, \lceil x^*_i \rceil - (2\Delta+1)^m\}$ and $u'_i = \min\{u_i, \lfloor x^*_i \rfloor + (2\Delta+1)^m\}$ for $i = 1, 2, \dots, n$, we can replace ℓ, u by ℓ', u' without affecting feasibility. By shifting the solution, we can reduce the lower bound to zero and we arrive at the following ILP that is equivalent to the original one.

$$\begin{aligned} Ax &= b - A\ell' \\ x_i &\in \{0, 1, \dots, u'_i - \ell'_i\} \qquad\qquad \text{for all } i = 1, 2, \dots, n \end{aligned}$$

Note that $u'_i - \ell'_i \leq 2m(2m\Delta+1)^m$. Thus replacing each variable by $u'_i - \ell'_i$ binary variables we arrive at an ILP with $n' \leq 2m(2m\Delta+1)^m \cdot n \leq 2m(2m\Delta+1)^{2m}$ binary variables.

2.6 n-Fold Integer Linear Programming

In this section, we prove Theorem 5.

▶ **Theorem 5.** *For every $\delta > 0$, there is no algorithm with running time $2^{O(m^{3-\delta})}\operatorname{poly}(n)$ for n-Fold Integer Linear Programming when the maximum absolute entry is bounded by $\Delta = O(1)$, unless Hypothesis 3 is false.*

Consider the ILP

$$\begin{aligned} Ax &= b \\ x &\in \{0,1\}^n \end{aligned} \tag{9}$$

with $A \in \{-2^m, \dots, 2^m\}^{m \times n}$. We will show that this can be formulated as an equivalent n-Fold Integer Linear Program with parameter m and $\Delta = O(1)$. The reduction follows a similar idea as one in [10], which derives a lower bound based on Subset Sum. Note that if (9) had arbitrary variables (not necessarily binary) then we can use the reduction of the previous section to transform it into a binary one. For $m \geq 3$, define

$$B = \begin{pmatrix} 1 & 0 & & \cdots & 0 & 1 \\ 0 & & & \cdots & & 0 \\ 2 & -1 & & \cdots & & \\ & 2 & -1 & \cdots & & \vdots \\ & & & \ddots & & \\ & & & 2 & -1 & 0 \end{pmatrix} \in \{-1, 0, 1, 2\}^{m \times m}.$$

This matrix has the property that the system $Bx = (1, 0, \dots, 0)^\mathsf{T}, x \in \mathbb{Z}^m_{\geq 0}$ has exactly two solutions, namely $x = (0, \dots, 0, 1)$ and $x = (1, 2^1, 2^2, \dots, 2^{m-2}, 0)$. Our n-Fold Integer Linear Program has one block A'_i, B'_i for each column A_i of A. Matrix B'_i is defined as B above with right-hand side $b'^{(i)} = (1, 0, \dots, 0)^\mathsf{T}$. Matrix A'_i is derived from A_i as follows: consider a coefficient A_{ji}. We rewrite

$$A_{ji} = \lambda_0 \cdot 2^0 + \lambda_1 \cdot 2^1 + \cdots + \lambda_{m-2} \cdot 2^{m-2} \text{ with } \lambda \in \{-2, \dots, 2\}^{m-1}.$$

Such a λ exists since $|A_{ji}| \leq 2^m$. Then we set the jth row of A'_i as $(\lambda^\mathsf{T}, 0)$. Let $x^{(i)}$ be the variables corresponding to block A'_i, B'_i. By choice of $B'_i, b'^{(i)}$ there are exactly two settings of $x^{(i)}$ that satisfy $B'_i x^{(i)} = b'^{(i)}$, namely $x^{(i)} = (0, \dots, 0, 1)$ and $x^{(i)} = (2^0, 2^1, \dots, 2^{m-2}, 0)$. Thus $A'_i x^{(i)} \in \{(0, \dots, 0)^\mathsf{T}, A_i\}$. Hence, the n-Fold Integer Linear Program with $b^{(0)} = b$ is equivalent to (9).

3 Algorithm for few distinct variables

In this section, we prove Theorem 6.

▶ **Theorem 6.** *Consider the integer programming problem*

$$\begin{aligned} &\max\ c^\mathsf{T} x \\ &\quad Ax = b \\ &\ell_i \leq x_i \leq u_i \qquad\qquad i = 1, \dots, n \\ &\quad x \in \mathbb{Z}^n\ . \end{aligned} \tag{1}$$

Let Δ be an upper bound on the absolute value of entries of A. The problem (1) can be solved in time

$$n^{m+1} \cdot O(m\Delta)^m \cdot \log(\|u - \ell\|_\infty).$$

We assume without loss of generality that in (1) we have $\ell = (0, 0, \dots, 0)$. Note that otherwise, one may obtain an equivalent ILP with $\ell = (0, 0, \dots, 0)$, $u' = u - \ell$, and $b' = b - A\ell$. We first decompose each u_i into a sum of powers of two with the properties described in the following lemma. Such a construction is well-known in literature, see e.g. [18, 4, 16].

▶ **Lemma 18** (Cover). *For every integer $k \in \mathbb{N}$ and $h \geq \lfloor \log k \rfloor$, there exist integers $c_0(k), \dots, c_h(k) \in \{0, 1, 2\}$ such that*

$$\left\{ \sum_{i=0}^{h} 2^i x_i : 0 \leq x_i \leq c_i(k) \text{ for all } 0 \leq i \leq h \right\} = \{0, 1, \dots, k\}.$$

We call such a set $c_0(k), \dots, c_h(k)$ a cover *of k. This set can be found in polynomial time in h.*

Proof. For $n \in \mathbb{N}$, let $\mathrm{bin}_i(n) \in \{0, 1\}$ denote the ith bit of the binary representation of n. Let $B := 2^h - 1$ be the *bottom* bits, and let $T = k - B$ represent the *top* bits of k. Now, consider the sets

$$S_B := \left\{ \sum_{i=0}^{h-1} 2^i x_i : 0 \leq x_i \leq \mathrm{bin}_i(B) \text{ for all } 0 \leq i \leq h-1 \right\} \text{ and}$$

$$S_T := \left\{ \sum_{i=0}^{h} 2^i x_i : 0 \leq x_i \leq \mathrm{bin}_i(T) \text{ for all } 0 \leq i \leq h \right\}.$$

We will set $c_i(k) := \mathrm{bin}_i(B) + \mathrm{bin}_i(T) \in \{0,1,2\}$. Showing that these numbers are a cover of k is equivalent to

$$S_B \oplus S_T := \{b + t \mid b \in S_B, t \in S_T\} = \{0, 1, \ldots, k\}.$$

First, observe that $S_B = \{0, 1, \ldots, 2^h - 1\}$. Moreover, there is no gap of length at least 2^h in the set S_T, that is, for every nonzero $a \in S_T$, there exists a smaller $b \in S_T$ such that $a - b < 2^h$. This b can be constructed by flipping an arbitrary bit of a to 0. Therefore, $S_B \oplus S_T$ consists of consecutive numbers. The smallest number in $S_B \oplus S_T$ is clearly 0, and the largest is $B + T = k$. ◀

We set $h = \lfloor \log_2(\|u\|_\infty) \rfloor$. Using the lemma above, for each i, we decompose $u_i = c_0(u_i) \cdot 2^0 + c_1(u_i) \cdot 2^1 + \cdots + c_h(u_i) \cdot 2^h$ with its cover. We can now naturally rewrite (1) as

$$\begin{aligned}
&\max \sum_{i=0}^{h} 2^i c^\mathsf{T} z^{(i)} \\
&\sum_{i=0}^{h} 2^i A z^{(i)} = b \\
&0 \le z_i^{(k)} \le c_k(u_i) && k \in \{0, \ldots, h\} \\
&z^{(0)}, \ldots, z^{(h)} \in \mathbb{Z}^n \ .
\end{aligned}$$

Each $z^{(j)}$ produces a partial solution for some unknown right-hand side $b^{(j)} = 2^j A z^{(j)}$. We write $b^{(\le j)} = 2^0 A z^{(0)} + 2^1 A z^{(1)} + \cdots + 2^j A z^{(j)}$.

Our approach is now to compute, for $j = 0, 1, \ldots, h$ and for every potential value of $b^{(\le j)}$, a solution $z^{(0)}, z^{(1)}, \ldots, z^{(j)}$. To do this, we first reduce the search space of relevant vectors $b^{(\le j)}$.

▶ **Lemma 19.** *Consider $A \in \mathbb{Z}^{m \times n}$ and $b \in \mathbb{Z}^m$ with $\|A\|_\infty \le \Delta$. Suppose that $y = y^{(0)} \cdot 2^0 + y^{(1)} \cdot 2^1 + \cdots + y^{(h)} \cdot 2^h$ satisfies $Ay = b$ and $0 \le y^{(j)} \le u^{(j)}$ with $u^{(j)} \in \{0,1,2\}^n$ for each $j \in \{0, \ldots, \ell\}$. Then for $b^{(\le j)} = 2^0 A y^{(0)} + 2^1 A y^{(1)} + \cdots + 2^j A y^{(j)}$, we have*

$$b^{(\le j)} \equiv b \mod 2^{j+1}$$

and furthermore

$$b^{(\le j)} \in \{-2^{j+2} mn\Delta, \ldots, 2^{j+2} mn\Delta\}^m \ .$$

Proof. The first property follows from the observation that $b^{(>j)} := b - b^{(\le j)}$ is a vector with each component being a multiple of 2^{j+1}. Hence, $b^{(\le j)} \equiv b - b^{(>j)} \equiv b \mod 2^{j+1}$.

For the second property, observe that $\|y^{(h)}\|_1 \le \|u^{(h)}\|_1 \le 2n$ for each h. Therefore, $\|b^{(\le j)}\|_\infty \le \sum_{h=0}^{j} 2^{h+1} n\Delta \le 2^{j+2} nm\Delta$. ◀

We say that a vector $b' \in \mathbb{Z}^m$ is *relevant* for j if $b' \equiv b \mod 2^{j+1}$ and $\|b'\|_\infty \le 2^{j+2} mn\Delta$. Clearly, the following holds:

▷ Claim 20. For every j, the number of relevant vectors for j is $O(mn\Delta)^m$.

We will now iteratively compute solutions for all vectors relevant for $j+1$ from the solutions for all relevant vectors for j with $j \ge 0$. Note that this will immediately establish Theorem 6.

▶ **Lemma 21.** *Let $\mathcal{S}_j$ be a set of optimal solutions to (1) for all $b' \in \mathbb{Z}^m$ that are relevant for j. Then, in time $n^{m+1} \cdot O(m\Delta)^m$, we can compute a set $\mathcal{S}_{j+1}$ of optimal solutions for all $b'' \in \mathbb{Z}^m$ relevant for $j+1$.*

Proof. Let V be the set of all vectors $b' \in \mathbb{Z}^m$ with $b' \equiv b \mod 2^{j+1}$ and $\|b'\|_\infty \leq 2^{j+2}mn\Delta$.

We define an edge-weighted directed acyclic graph with vertices $\{s\} \cup V^{(0)} \cup V^{(1)} \cup \ldots \cup V^{(n)}$, where $V^{(0)}, V^{(1)}, \ldots, V^{(n)}$ are $n+1$ copies of V, and s is a distinguished source vertex.

If $j \geq 0$, there is an edge from s to every vertex $v_{b'} \in V^{(0)}$ such that the vector $v_{b'}$ corresponds to a relevant vector $b' \in \mathbb{Z}^m$ for j for which (1) has a solution $x_{b'}$. This edge indicates feasibility, and the weight of the edge from s to $v_{b'}$ is the value $c^\mathsf{T} x_{b'}$ for the optimal solution $x_{b'}$. In the base case where $j < 0$, there is exactly one edge of weight 0 to the vertex in $V^{(0)}$ that corresponds to the all-zero vector.

For each vertex in $V^{(i-1)}$ for $0 < i \leq n$, there is an edge to the corresponding vertex in $V^{(i)}$ with weight zero. Further, if $c_{j+1}(u_i) \geq 1$, for every vertex corresponding to b' in $V^{(i-1)}$, there is an edge to the vertex in $V^{(i)}$ that corresponds to $b' + 2^j A_i$ (if it exists). The weight of this edge is $2^j c_i$.

Similarly, if $c_{j+1}(u_i) \geq 2$, then for every vertex in $V^{(i-1)}$ that corresponds to a relevant b', there is an edge to a vertex in $V^{(i)}$ that corresponds to $b' + 2^{j+1} A_i$ (if it exists), with a cost of $2^{j+1} c_i$.

Finally, we compute the longest path from s to each vertex in $V^{(n)}$, and we store these as the values of the solutions to all the relevant right-hand sides for $j+1$.

For the running time, observe that by Claim 20, we have $|V| \leq O(mn\Delta)^m$. Hence, the graph has $n^{m+1} \cdot O(m\Delta)^m$ vertices and edges. Thus, the longest path problem can be solved in time $n^{m+1} \cdot O(m\Delta)^m$.

For correctness, consider a path in the graph from vertex $v_{b_1} \in V^{(0)}$ to vertex $v_{b_2} \in V^{(n)}$ (corresponding to vectors b_1 and b_2 respectively). The edges of this path define a vector $z^{(j+1)}$ such that $0 \leq z^{(j+1)} \leq c_{j+1}(u)$. Moreover, by construction, it holds that $2^j A z^{(j+1)} + b_1 = b_2$. Finally, the weight of this path corresponds to the value $2^j c^\mathsf{T} z^{(j+1)}$. ◀

References

1. 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. doi:10.1145/3450524.
2. Karl Bringmann. Fine-grained complexity theory. In *Proceedings of STACS*, page 1, 2019.
3. Moses Charikar, Alantha Newman, and Aleksandar Nikolov. Tight Hardness Results for Minimizing Discrepancy. In Dana Randall, editor, *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011*, pages 1607–1614. SIAM, 2011. doi:10.1137/1.9781611973082.124.
4. Lin Chen, Jiayi Lian, Yuchen Mao, and Guochuan Zhang. Faster algorithms for bounded knapsack and bounded subset sum via fine-grained proximity results. In *Proceedings of SODA*, pages 4828–4848, 2024. doi:10.1137/1.9781611977912.171.
5. 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 SODA*, pages 1666–1681, 2021. doi:10.1137/1.9781611976465.101.
6. Marek Cygan, Holger Dell, Daniel Lokshtanov, Dániel Marx, Jesper Nederlof, Yoshio Okamoto, Ramamohan Paturi, Saket Saurabh, and Magnus Wahlström. On problems as hard as CNF-SAT. *ACM Transactions on Algorithms (TALG)*, 12(3):1–24, 2016. doi:10.1145/2925416.
7. Daniel Dadush, Arthur Léonard, Lars Rohwedder, and José Verschae. Optimizing low dimensional functions over the integers. In *Proceedings of IPCO*, pages 115–126, 2023. doi:10.1007/978-3-031-32726-1_9.
8. Friedrich Eisenbrand, Lars Rohwedder, and Karol Węgrzycki. Sensitivity, Proximity and FPT Algorithms for Exact Matroid Problems. In *2024 IEEE 65th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1610–1620, 2024. doi:10.1109/FOCS61266.2024.00100.

9 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. doi:10.1145/3340322.

10 Christoph Hunkenschröder, Kim-Manuel Klein, Martin Koutecký, Alexandra Lassota, and Asaf Levin. Tight lower bounds for block-structured integer programs. In *Proceedings of IPCO*, pages 224–237, 2024. doi:10.1007/978-3-031-59835-7_17.

11 Klaus Jansen and Lars Rohwedder. On integer programming, discrepancy, and convolution. *Mathematics of Operations Research*, 48(3):1481–1495, 2023. doi:10.1287/MOOR.2022.1308.

12 Ravi Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of operations research*, 12(3):415–440, 1987. doi:10.1287/MOOR.12.3.415.

13 Dušan Knop, Martin Koutecký, and Matthias Mnich. Combinatorial n-fold integer programming and applications. *Mathematical Programming*, 184(1):1–34, 2020. doi:10.1007/S10107-019-01402-2.

14 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. doi:10.1145/3397484.

15 Hendrik W. Lenstra Jr. Integer programming with a fixed number of variables. *Mathematics of operations research*, 8(4):538–548, 1983. doi:10.1287/MOOR.8.4.538.

16 Silvano Martello and Paolo Toth. *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc., 1990.

17 Christos H. Papadimitriou. On the complexity of integer programming. *Journal of the ACM (JACM)*, 28(4):765–768, 1981. doi:10.1145/322276.322287.

18 Adam Polak, Lars Rohwedder, and Karol Węgrzycki. Knapsack and subset sum with small items. In *Proceedings of ICALP*, pages 1–19, 2021. doi:10.4230/LIPICS.ICALP.2021.106.

19 Victor Reis and Thomas Rothvoss. The subspace flatness conjecture and faster integer programming. In *Proceedings of FOCS*, pages 974–988, 2023. doi:10.1109/FOCS57990.2023.00060.

20 Virginia Vassilevska Williams. Hardness of easy problems: Basing hardness on popular conjectures such as the strong exponential time hypothesis (invited talk). In *Proceedings of IPEC*, 2015. doi:10.4230/LIPIcs.IPEC.2015.17.