

INSTITUT FÜR INFORMATIK

Approximation Algorithms for Integer Programming with Resource Augmentation

Hauke Brinkop, Hua Chen, Lin Chen, Klaus Jansen,
Guochuan Zhang

Bericht Nr. 2502

September 2025

ISSN 2192-6247



CHRISTIAN-ALBRECHTS-UNIVERSITÄT
ZU KIEL

Institut für Informatik der
Christian-Albrechts-Universität zu Kiel
Olshausenstr. 40
D – 24098 Kiel

Approximation Algorithms for Integer Programming with Resource Augmentation

Hauke Brinkop, Hua Chen, Lin Chen, Klaus Jansen,
Guochuan Zhang

Bericht Nr. 2502
September 2025
ISSN 2192-6247

e-mail: {hab, kj}@informatik.uni-kiel.de,
chenhua_by@zju.edu.cn, chenlin198662@zju.edu.cn,
zgc@zju.edu.cn

Technical Report

Approximation Algorithms for Integer Programming with Resource Augmentation

Hauke Brinkop ✉ 

Kiel University

Lin Chen ✉ 


Zhejiang University

Hua Chen ✉ 

Zhejiang University

Klaus Jansen ✉ 

Kiel University

Guochuan Zhang ✉ 

Zhejiang University

Abstract

Solving a general integer program (IP) is NP-hard. The classic algorithm [Papadimitriou, J.ACM '81] for IPs has a running time depending on the number of constraints (m) and the largest absolute value of the entries of the constraint matrix (Δ). The running time is exponential in m , and becomes pseudo-polynomial if m is a constant. In recent years, there has been extensive research on FPT (fixed parameter tractable) algorithms for the so-called n -fold IPs, which may possess an arbitrary number of constraints, but the constraint matrix satisfies a specific block structure. It is remarkable that these FPT algorithms take Δ as a parameter, together with other parameters including the number of rows/columns of some small submatrices. If Δ is not treated as a parameter, then the running time becomes pseudo-polynomial even if all the other parameters are taken as constants.

This paper explores the trade-off between time and accuracy in solving an IP. We show that, for arbitrary $\varepsilon > 0$, there exists an algorithm that runs in $f(m, \varepsilon) \text{poly}(|I|)$ time, and returns a near-feasible solution that violates the constraints by at most $\varepsilon\Delta$. Furthermore, for n -fold IPs, we establish a similar result – our algorithm runs in time that depends on the number of rows and columns of small submatrices together with $1/\varepsilon$, and returns a solution that slightly violates the constraints. We show that our results can be used to obtain additive approximation schemes for multidimensional knapsack as well as scheduling.

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

Keywords and phrases Approximation algorithms, Resource augmentation, Integer programs, n -fold IPs

Acknowledgements This work was partially supported by the DFG project ‘Strukturaussagen für ganzzahlige lineare Programme’ (DFG JA 612 / 27-1)

1 Introduction

Integer programs (IPs) find widespread applications in many combinatorial optimization problems. Generally, an integer program can be formulated as follows:

$$\min \{ \mathbf{w}\mathbf{x} : H\mathbf{x} = \mathbf{b}, \ell \leq \mathbf{x} \leq \mathbf{u}, \mathbf{x} \in \mathbb{Z}^n \}, \quad (1)$$

where $H \in \mathbb{Z}^{m \times n}$ is the constraint matrix, $\mathbf{b} \in \mathbb{Z}^m$ and $\mathbf{w}, \ell, \mathbf{u} \in \mathbb{Z}^n$. Karp [41] showed that IP (1) is NP-hard in 1972. Subsequent studies thus focus on finding efficient algorithms for IPs where the constraint matrix has some specific structure.

Lenstra [39] studied IPs with few variables and presented a $2^{\mathcal{O}(n^3)} \cdot \text{poly}(|I|)$ -time algorithm, where $|I|$ denotes the length of the input. The running time has been improved in several

subsequent works [15, 18, 40, 52]. Very recently, Reis and Rothvoss [52] proved that IP (1) can be solved in $(\log n)^{\mathcal{O}(n)} \cdot \text{poly}(|I|)$ -time.

Papadimitriou [50] considered IPs with few constraints and provided an algorithm running in $n^{\mathcal{O}(m)}(m \cdot \max\{\Delta, \|\mathbf{b}\|_\infty\})^{\mathcal{O}(m^2)}$ -time, where $\Delta := \|H\|_\infty$ and $\|\mathbf{b}\|_\infty$ represent the largest absolute values among all entries of H and \mathbf{b} , respectively. The running time was later improved by Eisenbrand and Weismantel [24], and Jansen and Rohwedder [37]. In particular, for the special case when $\ell = \mathbf{0}$ and $\mathbf{u} = \infty$, IP (1) can be solved in $(m\Delta)^{\mathcal{O}(m)} \cdot \text{poly}(|I|)$ time [37], and such a running time is essentially the best possible assuming Exponential Time Hypothesis (ETH) [45].

The results mentioned above assume that the constraint matrix H has either few rows or few columns. There has also been extensive research on IPs where the number of rows and columns of H can be arbitrary, but H possesses some block structure, see, e.g., [2, 13, 14, 19, 23, 28, 29, 33, 42, 46]. One most widely studied structure is the so-called n -fold, where H consists of small submatrices A^i , D^i of the following form

$$H := \begin{pmatrix} D^1 & D^2 & \dots & D^n \\ A^1 & 0 & & 0 \\ 0 & A^2 & & 0 \\ \vdots & & \ddots & \\ 0 & 0 & & A^n \end{pmatrix},$$

where $A^i \in \mathbb{Z}^{s_A \times t}$, and $D^i \in \mathbb{Z}^{s_D \times t}$ for all i . Many parameterized algorithms for n -fold IPs have been developed [1, 12, 14, 22, 23, 28, 34, 35]. In particular, Cslovjcek et al. [12] presented an algorithm of running time $2^{\mathcal{O}(s_A^2 s_D)} (s_D s_A \Delta)^{\mathcal{O}(s_A^2 + s_A s_D^2)} \cdot (nt)^{1+o(1)}$, where $\Delta = \|H\|_\infty$.

We see that the running time is either exponential in the number of variables or dependent on Δ and is thus pseudo-polynomial. Such a situation is unavoidable if we are looking for an exact algorithm that solves IP (1) optimally. However, a polynomial time algorithm may be possible if we are allowed to solve IP (1) approximately in the sense that the constraints can be violated slightly.

One most well-known example is Knapsack, where H contains one row, i.e., $m = 1$, $\ell = \mathbf{0}$, $\mathbf{u} = \mathbf{1}$, and all entries of \mathbf{w} and H are nonnegative¹. Knapsack admits an FPTAS whose running time is polynomial in the input length $|I|$ and $1/\varepsilon$ instead of Δ (see, e.g., [6, 11, 20, 30, 38, 47]). Alternatively, it admits an algorithm of the same running time that returns a solution whose objective value is at least OPT (where OPT denotes the optimal objective value satisfying the constraint), while violating the constraint by a factor of $1 + \varepsilon$.

Another example is due to Dadush [16], who presented an algorithm of running time $2^{\mathcal{O}(n)}(1/\varepsilon^2)^n$ which either asserts that the convex body K described by the constraints does not contain any integer points, or finds an integer point in the body stemming from K scaled by $1 + \varepsilon$ from its center of gravity.

Motivated by the above two examples, this paper is aimed at approximation algorithms for IPs. We are interested in an algorithm that returns a near-feasible solution that may violate the constraints by a factor of $1 + \varepsilon$, but runs in polynomial time in the input. Such relaxation of constraints is also referred to as *resource augmentation*, which has received much attention in various combinatorial optimization problems including Knapsack [4, 32], Subset Sum [10, 48], Square Packing [25], etc.

¹ Knapsack requires $H\mathbf{x} \leq \mathbf{b}$ instead of $H\mathbf{x} = \mathbf{b}$, but we can add dummy items of 0 profit and unit weight to enforce the equation.

1.1 Our contribution

Our first result is an approximation scheme for IPs with few constraints.

► **Theorem 1.** *Given is an integer program IP (1) with optimal objective value OPT , where $H \in \mathbb{Z}^{m \times n}$. Let $\varepsilon > 0$ be an arbitrarily small constant. Then there exists an algorithm of running time $f(m, \varepsilon) \text{poly}(|I|)$ which returns a near-feasible solution $\tilde{\mathbf{x}}$ such that $\tilde{\mathbf{x}} \in \{\mathbf{x} : \|H\mathbf{x} - \mathbf{b}\|_\infty \leq \varepsilon\Delta, \ell \leq \mathbf{x} \leq \mathbf{u}, \mathbf{x} \in \mathbb{Z}^n\}$, and $\mathbf{w}\tilde{\mathbf{x}} \leq \text{OPT}$.*

We remark that OPT always refers to the optimal objective value **without** violating the constraints.

Theorem 1 extends Papadimitriou's algorithm in the direction of approximation. Here we may interpret IP (1) as a generalized multidimensional knapsack problem where items may have negative profits or (multidimensional) weights. In such a setting, a multiplicative $(1 + \varepsilon)$ -approximation may not be possible, since the sum of profits or weights can become 0. Nevertheless, Theorem 1 shows that an additive error with respect to the largest absolute value of the weight is always achievable.

We then extend our result to n -fold IPs and obtain the following.

► **Theorem 2.** *Given is an integer program $\min\{\mathbf{w}\mathbf{x} : \sum_{i=1}^n D^i \mathbf{x}^i = \mathbf{b}^0, A^i \mathbf{x}^i = \mathbf{b}^i, 1 \leq i \leq n, \mathbf{0} \leq \mathbf{x} \leq \mathbf{u}, \mathbf{x} \in \mathbb{Z}^{nt}\}$ with optimal objective value OPT , where $A^i \in \mathbb{Q}_{\geq 0}^{s_A \times t_A}$, $D^i \in \mathbb{Q}_{\geq 0}^{s_D \times t_D}$, and $t_A = t_D = t$. Let $\varepsilon > 0$ be an arbitrarily small constant. Then there exists an algorithm of running time $f(s_A, s_D, t, \varepsilon) \text{poly}(|I|)$ which returns a near-feasible solution $\tilde{\mathbf{x}}$ such that $\tilde{\mathbf{x}} \in \{\mathbf{x} : (1 - \varepsilon)\mathbf{b}^0 \leq \sum_{i=1}^n D^i \mathbf{x}^i \leq (1 + \varepsilon)\mathbf{b}^0, (1 - \varepsilon)\mathbf{b}^i \leq A^i \mathbf{x}^i \leq (1 + \varepsilon)\mathbf{b}^i, 1 \leq i \leq n, \mathbf{0} \leq \mathbf{x} \leq \mathbf{u}, \mathbf{x} \in \mathbb{Z}^{nt}\}$, and $\mathbf{w}\tilde{\mathbf{x}} \leq \text{OPT}$.*

n -fold IPs have been used to model a variety of combinatorial optimization problems, including, e.g., problems in computational social choice [44] and scheduling [43]. The FPT algorithms for n -fold IPs are used to derive FPT algorithms for these problems. Our Theorem 2 may be utilized to develop approximation schemes for relevant problems.

Towards proving Theorem 2, we shall first show the following theorem.

► **Theorem 3.** *Given is an integer program $\min\{\mathbf{w}\mathbf{x} : \sum_{i=1}^n D^i \mathbf{x}^i = \mathbf{b}^0, \mathbf{x}^i \in \mathcal{P}^i, 1 \leq i \leq n, \mathbf{x} \in \mathbb{Z}^{nt}\}$ with optimal objective value OPT , where $D^i \in \mathbb{Q}^{s \times t}$ and \mathcal{P}^i is an arbitrary set of integer vectors. Let $\varepsilon > 0$ be an arbitrarily small constant, and $\kappa = \max_{\mathbf{u} \in \cup_{i=1}^n \mathcal{P}^i} \|\mathbf{u}\|_\infty$, i.e., κ is the largest ℓ_∞ -norm among all integer vectors in $\cup_{i=1}^n \mathcal{P}^i$. Then there exists an algorithm of running time $f(s, t, \kappa, \varepsilon) \text{poly}(|I|)$ which returns a near-feasible solution $\tilde{\mathbf{x}}$ such that $\tilde{\mathbf{x}} \in \{\mathbf{x} : \|\sum_{i=1}^n D^i \mathbf{x}^i - \mathbf{b}^0\|_\infty \leq \varepsilon\Delta, \mathbf{x}^i \in \mathcal{P}^i, 1 \leq i \leq n, \mathbf{x} \in \mathbb{Z}^{nt}\}$, and $\mathbf{w}\tilde{\mathbf{x}} \leq \text{OPT}$, where $\Delta = \max_{i \in [n]} \|D^i\|_\infty$.*

Theorem 3 may be of separate interest. One application is to provide additive approximation schemes for the unrelated machine scheduling problem: given are n jobs and s machines, with the processing time of job i on machine h as p_{ih} . The goal is to assign jobs to machines such that the makespan is at most some target value T . Let $x_h^i \in \{0, 1\}$ denote whether job i is assigned to machine h , i.e., if job i is assigned to machine h , then $x_h^i = 1$; otherwise, $x_h^i = 0$. Then the natural IP formulation is the feasibility test:

$$\left\{ \mathbf{x} : \sum_{i=1}^n p_{ih} x_h^i \leq T, \forall 1 \leq h \leq s, \mathbf{x}^i \in \mathcal{P}^i, 1 \leq i \leq n, \mathbf{x} \in \mathbb{Z}^{nt} \right\},$$

where polytope $\mathcal{P}^i := \{\mathbf{x}^i : x_1^i + x_2^i + \dots + x_s^i = 1, x_h^i \in \{0, 1\}\}$ implies that exactly one x_h^i is 1. Observe that $D^i \in \mathbb{Z}^{s \times s}$ and $\kappa = 1$, so Theorem 3 implies a PTAS of running time $f(s, \varepsilon) n^{\mathcal{O}(1)}$ for finding a feasible schedule $\tilde{\mathbf{x}}$ such that the makespan is at most $T + \varepsilon \cdot \max_{i \in [n], h \in [s]} p_{ih}$.

In comparison, the current best-known PTAS for unrelated machine scheduling runs in $2^{\mathcal{O}(s \log(s/\delta))}$ -time [36], and returns a solution with makespan $(1 + \delta)T$. Using this algorithm to derive an additive PTAS would require setting $\delta = \mathcal{O}(\varepsilon/n)$, yielding a non-polynomial running time.

We remark that additive approximation schemes have been studied for other scheduling problems. In particular, Buchem et al. [5] presented an algorithm for $P||C_{\max}$ with a running time of $s^2 n^{\mathcal{O}(1/\varepsilon)}$ that computes a solution with makespan at most $T + \varepsilon \cdot \max_{j \in [n]} p_j$.

1.2 Technical overview

We first briefly discuss our technique for proving Theorem 1. Let $H = (\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_n)$, whereas $H\mathbf{x} = \sum_{j=1}^n \mathbf{h}_j x_j$. Observe that $\mathbf{h}_j \in [-\Delta, \Delta]^m$. If n is small, say $n = m^{\mathcal{O}(1)}$, then Lenstra's algorithm [39] works directly and solves the IP in $n^{\mathcal{O}(n)} = 2^{m^{\mathcal{O}(1)}}$ time, concluding Theorem 1. So we may assume that n is sufficiently large, which means there are sufficiently many \mathbf{h}_j 's within the box $[-\Delta, \Delta]^m$. Consequently, many \mathbf{h}_j 's are very similar to each other in the sense that their difference lies in a small hypercube of side length $\mathcal{O}(\varepsilon\Delta)$. Now suppose \mathbf{h}_1 and \mathbf{h}_2 are similar. To approximately solve IP (1), we observe that it is not necessary to keep two integer variables x_1 and x_2 for \mathbf{h}_1 and \mathbf{h}_2 respectively. Instead, we may introduce a mega integer variable y that represents $x_1 + x_2$, and relax x_1 and x_2 to fractional variables. By doing so, we transform the original IP to a mixed integer program (MIP) with a constant number of integer variables, which is solvable in polynomial time. Then we round fractional variables to integer variables by maintaining the overall error incurred. The challenge here is that, to bound the overall error, we need to make sure that among all the fractional variables, only a few of them can take fractional values. Interestingly, while these fractional variables need to satisfy a large number of constraints, the specific structure of the constraint matrix allows us to conclude that very few variables may take a fractional value in a vertex solution. Towards showing this, we introduce some necessary notions and lemmas regarding vertex solutions in preliminaries.

The proof strategy for Theorem 2 can be viewed as a combination of that for Theorem 1 and Theorem 3, so we briefly discuss our technique for proving Theorem 3. For simplicity, let us assume that $\mathcal{P}^i = \mathcal{P} = \{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_\tau\}$ for all i . Hence, we see that the vector $D^i \mathbf{x}^i$ is essentially one column out of the precomputed matrix $\mathcal{D}^i := \{D^i \mathbf{p}_1, \dots, D^i \mathbf{p}_\tau\}$. If D^i and D^j (and hence \mathcal{D}^i and \mathcal{D}^j) are close to each other, then we do not want to have separate integer variables \mathbf{x}^i and \mathbf{x}^j for them, instead, we want to introduce a mega integer variable \mathbf{y} representing $\mathbf{x}^i + \mathbf{x}^j$, and relax \mathbf{x}^i and \mathbf{x}^j to fractional variables. It turns out that we need to go a bit further: we need τ mega integer variables \mathbf{y}_1 to \mathbf{y}_τ since there are τ choices out of \mathcal{P}^i . Going a bit further does not affect much when we transform the original IP into an MIP with a constant number of integer variables; however, it causes the established MIP to contain way more constraints. Our next goal is still to argue that in a vertex solution of the LP (induced by the MIP by fixing the integer variables), only a few variables may take a fractional value. To achieve this, we observe that the constraint matrix of the LP can be decomposed into two parts (two submatrices), one having a small rank, and the other having a huge rank but also having a nice diagonal block structure. We show that the property of the vertex solution to such a two-part LP can be studied by arguing on the two parts separately.

1.3 Related work

The approximation algorithms of IPs are rare in the literature. Dadush [16] studied *approximate integer programming* in 2014. Later, Dadush et al. [17] investigated how to reduce the exact integer programming problem to the approximate version. In addition, Shmoys and Swamy [53] showed that one could derive approximation algorithms for most of the stochastic IPs considered in [21, 26, 31, 51] by adopting a natural LP rounding approach. Swamy and Shmoys [54] gave the first approximation algorithms for a variety of k -stage generalizations of basic combinatorial optimization problems including the set cover, vertex cover, multicut on trees, facility location, and multicommodity flow problems.

Parameterized algorithms also exist for other block-structured IPs, such as two-stage stochastic IPs. Pioneering work on the IPs was conducted by Hemmecke and Schultz [29], followed by a series of subsequent improvements on the running times, e.g., [2, 13, 23, 33, 42, 46]. In particular, the current best-known result is due to Cslovjecsek et al. [13] with running time $2^{(2\Delta)^{\mathcal{O}(t_B(t_A+t_B))}}} \cdot n \log^{\mathcal{O}(t_A t_B)} n$. However, for 4-block n -fold IPs, whether an FPT algorithm exists remains unknown. Recently, Oertel et al. [49] presented an algorithm running in $n^{\mathcal{O}(\min\{t_B+2, s_D\}t_B)}$ time. Besides, there are some other results about 4-block n -fold IPs, e.g., [7–9, 14, 27]. The constraint matrices for the two IPs are shown as follows:

$$H_{\text{two-stage}} := \begin{pmatrix} B^1 & A^1 & 0 & & 0 \\ B^2 & 0 & A^2 & & 0 \\ \vdots & & & \ddots & \\ B^n & 0 & 0 & & A^n \end{pmatrix}, \quad H_{\text{4-block}} := \begin{pmatrix} C & D^1 & D^2 & \cdots & D^n \\ B^1 & A^1 & 0 & & 0 \\ B^2 & 0 & A^2 & & 0 \\ \vdots & & & \ddots & \\ B^n & 0 & 0 & & A^n \end{pmatrix}.$$

1.4 Preliminaries

Notations. We write column vectors in boldface, e.g., \mathbf{x}, \mathbf{y} , and their entries in normal font, e.g., x_j, y_j . Let $[i] := \{1, 2, \dots, i\}$. For a vector or a matrix, let $\|\cdot\|_\infty$ denote the maximum absolute value of its elements. Let $\text{poly}(|I|)$ denote a polynomial in the input length $|I|$, f be a computable function, and OPT denote the optimal objective value.

The support of a vector \mathbf{x} is the set of indices $I = \text{supp}(\mathbf{x})$ such that $x_j = 0 \iff j \notin I$. Let V and W be vector spaces and $\phi: V \rightarrow W$. The kernel of ϕ is the set of vectors mapping to zero, i.e., $\ker(\phi) = \{x \in V : \phi(x) = \mathbf{0}\}$. The nullity of ϕ , denoted $\text{nullity}(\phi)$, is the dimension of its kernel, that is, the dimension of the smallest dimensional vector space containing $\ker(\phi)$. The image of ϕ is defined as $\text{img}(\phi) = \{\phi(x) : x \in V\} \subseteq W$. The rank of ϕ , denoted $\text{rank}(\phi)$, is, analogously to the nullity, the dimension of the smallest dimensional vector space containing $\text{img}(\phi)$. If ϕ is a linear map, the definitions of rank and nullity simplify to

$$\text{nullity}(\phi) = \dim(\ker(\phi)), \quad \text{rank}(\phi) = \dim(\text{img}(\phi)).$$

The definitions are also used for matrices as a matrix $A \in \mathbb{R}^{m \times n}$ represents the linear map $\phi: \mathbb{R}^n \rightarrow \mathbb{R}^m, \mathbf{x} \mapsto A\mathbf{x}$. The kernel of a matrix $A \in \mathbb{R}^{m \times n}$ is $\ker(A) = \{\mathbf{x} \in \mathbb{R}^n : A\mathbf{x} = \mathbf{0}\}$, the nullity is its dimension, the image is $\{A\mathbf{x} \in \mathbb{R}^m : \mathbf{x} \in \mathbb{R}^n\}$ and the rank is its dimension. The latter coincides with the alternative rank definition of matrices as the largest number k such that any k columns of A are linearly independent.

We make use of the well-known rank-nullity theorem, stating that a linear map splits the dimension of the domain into the dimension of the kernel and the dimension of the image (cf. the splitting lemma).

► **Theorem 4** (Rank-Nullity Theorem). *Let V, W be vector spaces, where $\dim(V) < \infty$, and $\phi: V \rightarrow W$ a linear map. Then*

$$\dim(V) = \text{nullity}(\phi) + \text{rank}(\phi) .$$

We are generally interested in the number of variables in an LP that attain non-integral values. Thus, analogously to the support of a vector, we define the non-integral support.

► **Definition 5** (Non-Integral Support).

$$\text{nisupp}(\mathbf{x}) := \{j \in \text{supp}(\mathbf{x}) : x_j \notin \mathbb{Z}\} .$$

We will need the following proposition, which generalizes the fact that for a system $A\mathbf{x} = \mathbf{b}$, $\mathbf{x} \geq 0$, the support of a vertex solution consists of linearly independent columns, to systems with (additional) lower and upper bounds on the variables:

► **Proposition 6** (Proposition 3.3.3, [3]). *Let \mathbf{x}^* be a vertex solution to an LP of the form $A\mathbf{x} = \mathbf{b}$, $\ell \leq \mathbf{x} \leq \mathbf{u}$, $\ell, \mathbf{u} \in \mathbb{R}^m$. Then $(A_j)_{j \in \text{supp}(\mathbf{x}^*) : \ell_j < x_j < u_j}$ is non-singular.*

Given that a vertex solution can be computed in polynomial time, the following lemma is straightforward via Proposition 6.

► **Lemma 7.** *Consider an LP with the objective of minimizing or maximizing $\mathbf{w}\mathbf{x}$ subject to $\{\mathbf{x} : A\mathbf{x} = \mathbf{b}, \ell \leq \mathbf{x} \leq \mathbf{u}, \ell, \mathbf{u} \in \mathbb{Z}^n\}$, with $\text{OPT} \in \mathbb{R}$. In polynomial time we can compute an optimal solution \mathbf{x}^* such that $(A_j)_{j \in \text{supp}(\mathbf{x}^*) : \ell_j < x_j < u_j}$ is non-singular.*

2 Approximation algorithms for IPs with a constant number of constraints - Proof of Theorem 1

This section is devoted to the proof of Theorem 1. We follow the general method we described in “Technical overview”. The whole proof is divided into three steps.

Step 1: Transform IP (1) to a mixed IP with $\mathcal{O}(1)$ integer variables.

Recall that $H = (\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_n)$ where $\mathbf{h}_j \in [-\Delta, \Delta]^m$. We cut $[-\Delta, \Delta]^m$ into $(2/\delta)^m$ small boxes (called δ -boxes), $\prod_{i=1}^m [(\lambda_i - 1)\delta\Delta, \lambda_i\delta\Delta]$, where $\lambda_i \in \{-1/\delta + 1, -1/\delta + 2, \dots, 1/\delta\}$. We index all the δ -boxes arbitrarily, and let I_k be the set of indices j 's such that \mathbf{h}_j is in the k -th δ -box, where $k = 1, \dots, (2/\delta)^m$. Note that if some \mathbf{h}_j belongs to multiple δ -boxes, let it be in arbitrary one such δ -box. For the k -th δ -box, we pick one fixed vector \mathbf{v}_k and call it the *canonical vector*. For each $j \in I_k$, we define $\tilde{\mathbf{h}}_j = \mathbf{h}_j - \mathbf{v}_k$, so it is clear that $\|\tilde{\mathbf{h}}_j\|_\infty \leq \delta\Delta$.

We say that the vectors in the same δ -box are similar. By introducing a new variable y_k to the k -th δ -box, where $y_k := \sum_{j \in I_k} x_j$, we observe that $H\mathbf{x} = \sum_{j=1}^n \mathbf{h}_j x_j$ can be rewritten as $\sum_{k=1}^{(2/\delta)^m} \sum_{j \in I_k} (\mathbf{v}_k + \tilde{\mathbf{h}}_j) x_j$, thus obtaining the following MIP:

$$(\text{MIP}_1) \quad \min \quad \sum_{j=1}^n w_j x_j \tag{2a}$$

$$\sum_{k=1}^{(2/\delta)^m} \mathbf{v}_k y_k + \sum_{k=1}^{(2/\delta)^m} \sum_{j \in I_k} \tilde{\mathbf{h}}_j x_j = \mathbf{b} \tag{2b}$$

$$\sum_{j \in I_k} x_j = y_k, \quad \forall 1 \leq k \leq (2/\delta)^m \tag{2c}$$

$$\ell_j \leq x_j \leq u_j, \quad \forall j \in I_k, 1 \leq k \leq (2/\delta)^m \tag{2d}$$

$$y_k \in \mathbb{Z}, x_j \in \mathbb{R}, \quad \forall 1 \leq j \leq n, 1 \leq k \leq (2/\delta)^m \tag{2e}$$

Notice that (MIP₁) relaxes $x_j \in \mathbb{Z}$ to $x_j \in \mathbb{R}$, and only contains $(2/\delta)^m$ integer variables, y_k . Hence, applying Kannan's algorithm [40], an optimal solution to (MIP₁) can be computed in $(2/\delta)^{\mathcal{O}(m(2/\delta)^m)} \cdot |I|$ time.

Let \mathbf{y}^* and \mathbf{x}^* be the optimal solution to (MIP₁). In the following step, we will round fractional variables such that most fractional variables take integral values, and all constraints are still satisfied.

Step 2: Obtain a feasible solution with $\mathcal{O}(m)$ variables taking a fractional value.

Towards rounding x_j^* 's without changing the constraints in (MIP₁), it suffices to consider the following LP.

$$\begin{aligned}
 (\text{LP}_2) \quad \min \quad & \sum_{j=1}^n w_j x_j \\
 & \sum_{j \in I_k} \tilde{\mathbf{h}}_j x_j = \sum_{j \in I_k} \tilde{\mathbf{h}}_j x_j^*, \quad \forall 1 \leq k \leq (2/\delta)^m \quad (3a) \\
 & \sum_{j \in I_k} x_j = \sum_{j \in I_k} x_j^*, \quad \forall 1 \leq k \leq (2/\delta)^m \quad (3b) \\
 & l_j \leq x_j \leq u_j, \quad x_j \in \mathbb{R}, \quad \forall j \in I_k
 \end{aligned}$$

Let $\bar{\mathbf{x}} = (\bar{x}_j)_{j=1}^n$ be an optimal solution to the above (LP₂) computed via Lemma 7. We will argue on the number of variables in $\bar{\mathbf{x}}$ taking a fractional value. Towards this, let $\eta := |\text{nisupp}(\bar{\mathbf{x}})|$.

It is straightforward to show that $\eta \leq \mathcal{O}(\delta^{-m})$. However, this will not be sufficient to bound the rounding error. The following claim gives a sharper bound.

▷ **Claim 8.** $\eta \leq 2m$.

Proof. For simplicity, let the constraints of (LP₂) be $\mathbf{Ax} = \mathbf{b}$, while noting that \mathbf{b} is integral since \mathbf{y}^* is integral. Let $B = (A_j)_{j \in \text{nisupp}(\bar{\mathbf{x}})}$ be the submatrix induced by $\text{nisupp}(\bar{\mathbf{x}})$. Note that by Lemma 7, B is non-singular.

Observe that for every k we either have $|\text{nisupp}(\bar{\mathbf{x}}) \cap I_k| = 0$ or $|\text{nisupp}(\bar{\mathbf{x}}) \cap I_k| \geq 2$ due to Constraint (3b). If $|\text{nisupp}(\bar{\mathbf{x}}) \cap I_k| = 0$, Constraint (3b) becomes a zero-row in B . Let C be the matrix B after removing all zero-rows. Then C is non-singular (as we only removed zero rows from B). Moreover, since I_k 's are disjoint, there are at most $\eta/2$ constraints in Constraint (3b) which do not become a zero-row. Hence, there are at most $m + \frac{\eta}{2}$ constraints and η columns in C . If C has more columns than rows, then C is singular, which is a contradiction. So we have $m + \frac{\eta}{2} \geq \eta$, or equivalently $\eta \leq 2m$. ◁

Finally, we round the $\mathcal{O}(m)$ variables in $\bar{\mathbf{x}}$ that take fractional values.

Step 3: Round the $\mathcal{O}(m)$ fractional variables.

Let $I_k^* \subseteq I_k$ be the indices of variables that still take a fractional value after Step 2. Without loss of generality, we may assume that for $j \in I_k^*$, w_j 's are in non-decreasing order.

Let $\gamma := \sum_{j \in I_k^*} (\bar{x}_j - \lfloor \bar{x}_j \rfloor) = \sum_{j \in I_k^*} (\bar{x}_j - \lfloor \bar{x}_j \rfloor)$. Constraint (3b) guarantees that γ is an integer. Therefore, we can round up the first γ fractional variables in I_k^* (i.e., variables of the smallest indices) to $\lceil \bar{x}_j \rceil$, and meanwhile round down the other fractional variables in I_k^* to $\lfloor \bar{x}_j \rfloor$. This rounding method preserves the constraint $\sum_{j \in I_k} x_j = y_k$, and meanwhile does not increase the objective value. However, this rounding will introduce $\mathcal{O}(m\delta\Delta)$ error to Constraint (2b). Thus, Theorem 1 follows by taking $\delta = \mathcal{O}(\frac{\epsilon}{m})$.

Running time.

Step 1: $(2/\delta)^{\mathcal{O}(m(2/\delta)^m)} \cdot |I| = 2^{(\frac{m}{\epsilon})^{\mathcal{O}(m)}} \cdot |I|$

Step 2: $n^4 \cdot |I|$

Step 3: $(2/\delta)^m \cdot n \log n$

All in all, the total time is $2^{(\frac{m}{\varepsilon})^{O(m)}} \cdot \text{poly}(|I|)$, which has the form of $f(m, \varepsilon) \text{poly}(|I|)$.

3 Proof of Theorem 3

To prove Theorem 2, we first need to prove Theorem 3. We restate the theorem below.

► **Theorem 3.** *Given is an integer program $\min\{\mathbf{w}\mathbf{x} : \sum_{i=1}^n D^i \mathbf{x}^i = \mathbf{b}^0, \mathbf{x}^i \in \mathcal{P}^i, 1 \leq i \leq n, \mathbf{x} \in \mathbb{Z}^{nt}\}$ with optimal objective value OPT , where $D^i \in \mathbb{Q}^{s \times t}$ and \mathcal{P}^i is an arbitrary set of integer vectors. Let $\varepsilon > 0$ be an arbitrarily small constant, $\kappa = \max_{\mathbf{u} \in \cup_{i=1}^n \mathcal{P}^i} \|\mathbf{u}\|_\infty$, i.e., κ is the largest ℓ_∞ -norm among all integer vectors in $\cup_{i=1}^n \mathcal{P}^i$. Then there exists an algorithm of running time $f(s, t, \kappa, \varepsilon) \text{poly}(|I|)$ which returns a near-feasible solution $\tilde{\mathbf{x}}$ such that $\tilde{\mathbf{x}} \in \{\mathbf{x} : \|\sum_{i=1}^n D^i \mathbf{x}^i - \mathbf{b}^0\|_\infty \leq \varepsilon \Delta, \mathbf{x}^i \in \mathcal{P}^i, 1 \leq i \leq n, \mathbf{x} \in \mathbb{Z}^{nt}\}$, and $\mathbf{w}\tilde{\mathbf{x}} \leq \text{OPT}$, where $\Delta = \max_{i \in [n]} \|D^i\|_\infty$.*

To prove Theorem 3, we follow the same general strategy as the proof of Theorem 1. That is, we first reformulate the input IP into an MIP via some appropriate δ -boxes, and then we solve the MIP, fix the values of integer variables and obtain an LP, and argue that the vertex solution of this LP only contains a small number of variables taking a fractional value. Finally we round those fractional variables. The main challenge lies in the argument for the vertex solution, which builds upon a specific block structure of the constraint matrix. To facilitate this argument, we need to further introduce some notions.

3.1 General Mechanisms for LP that consists of two parts

We will encounter the case where the constraint matrix can be partitioned into two parts/sub-systems: One with small rank and one where for a given solution either the non-integral support is small or we can easily find linearly independent kernel elements acting only on the non-integral support. Assume we have an LP of the following form:

$$A\mathbf{x} = \mathbf{b}_A, \quad C\mathbf{x} = \mathbf{b}_C, \quad \mathbf{x} \geq \mathbf{0},$$

where A is a huge matrix of huge rank, but C has only a few, let's say ι , rows. Consider some fixed solution \mathbf{x}^* where we want to reduce $\text{nisupp}(\mathbf{x}^*)$ as much as possible. Now, since A has huge rank, the combined system also has huge rank, so it is not really helpful trying to argue via that. However, assume that we are able to find a set \mathcal{K} , $|\mathcal{K}| \geq \iota + 1$, of linearly independent non-trivial kernel elements of A , that are only non-zero on variables in $\text{nisupp}(\mathbf{x})$. Then we can “combine” those kernel elements to a non-trivial kernel element of the combined system, using Theorem 4.

More formally, following lemma then allows us to show non-zero nullity of the combined system if enough kernel elements of the first system (more than the rank of the second one) can be found.

► **Lemma 9.** *Consider vector spaces V, W where $\dim(V) < \infty$ and two linear maps $\phi_1, \phi_2: V \rightarrow W$. If $\text{nullity}(\phi_1) > \text{rank}(\phi_2)$, then $\dim(\ker(\phi_1) \cap \ker(\phi_2)) > 0$.*

Proof. Let $\psi: \ker(\phi_1) \rightarrow W$, $x \mapsto \phi_2(x)$ be the restriction of ϕ_2 to $\ker(\phi_1)$. Then $\text{rank}(\psi) \leq \text{rank}(\phi_2)$ and $\ker(\phi_1) \cap \ker(\phi_2) = \ker(\psi)$, and thus

$$\begin{array}{ccccc} & \text{Theorem 4} & \text{rank}(\psi) \leq \text{rank}(\phi_2) & \text{assumption} & \\ & \downarrow & \downarrow & \downarrow & \\ \dim(\ker(\phi_1) \cap \ker(\phi_2)) = \text{nullity}(\psi) & \stackrel{\text{Theorem 4}}{=} \dim(\ker(\phi_1)) - \text{rank}(\psi) & \geq \text{nullity}(\phi_1) - \text{rank}(\phi_2) & \stackrel{\text{assumption}}{>} 0 & . \blacktriangleleft \end{array}$$

The scenario mentioned above is then covered by applying this lemma with $V = \ker(A)$, $W = \text{img}(C)$, $\phi_1: \mathbf{x} \mapsto A\mathbf{x}$, $\phi_2: \mathbf{x} \mapsto C\mathbf{x}$: Given that $\text{nullity}(A) > \text{rank}(C)$, there is a non-trivial kernel element of the combined system. Note that $\text{nullity}(A) > \text{rank}(C)$ translates to “there are at least $\text{rank}(C) + 1$ linearly independent non-trivial kernel elements of A ”.

If the constraint matrix of an LP is totally unimodular and the right-hand side is integral, it is well-known that vertex solutions are integers.

► **Proposition 10.** *Let \mathbf{x} be a solution to an LP of the form $A\mathbf{x} = \mathbf{b}$, $\ell \leq \mathbf{x} \leq \mathbf{u}$, $\ell, \mathbf{u} \in \mathbb{R}^m$, $\mathbf{b} \in \mathbb{Z}^m$, where the constraint matrix A is totally unimodular. Then if $\text{nisupp}(\mathbf{x}) \neq \emptyset$, we have $\text{nullity}((A_j)_{j \in \text{nisupp}(\mathbf{x})}) > 0$.*

Proof. Let B be the matrix A restricted to the columns from $\text{nisupp}(\mathbf{x})$. Since A is totally unimodular, so is B . Let \mathbf{y} be the vector \mathbf{x} restricted to entries in $\text{nisupp}(\mathbf{x})$. Then $B\mathbf{y}$ is integral, as $A\mathbf{x}$ is integral and only x_j 's that are integral have been deleted from \mathbf{x} to obtain \mathbf{y} . Let $\gamma := A\mathbf{x} - B\mathbf{y}$. Then γ is an integer vector as $A\mathbf{x}$ and $B\mathbf{y}$ also are. Equivalently, $B\mathbf{y} = A\mathbf{x} - \gamma$. Consider the LP $B\mathbf{z} = A\mathbf{x} - \gamma$ with \mathbf{z} being variables. It is feasible (due to \mathbf{y}). Further, there is an integer vector \mathbf{y}^* such that $B\mathbf{y}^* = A\mathbf{x} - \gamma$, since B is totally unimodular. By construction, \mathbf{y} has no integral components, so $\mathbf{y} - \mathbf{y}^*$ is a non-trivial kernel element of B , directly implying B , and hence also A , has non-zero nullity. ◀

Rephrasing the above: a solution to a totally unimodular system with integral right-hand side is either integral or the subsystem induced by its non-integral variables is underdetermined.

3.2 Proof of Theorem 3

Now we are ready to prove Theorem 3. Let $\tau := \max_{i \in [n]} |\mathcal{P}^i| = \kappa^t$. By adding dummy vectors, we may assume $|\mathcal{P}^i| = \tau$ for all i and let $\mathcal{P}^i = \{\mathbf{p}_1^i, \mathbf{p}_2^i, \dots, \mathbf{p}_\tau^i\}$. Consider the matrix $\mathcal{D}^i = (D^i \mathbf{p}_1^i, D^i \mathbf{p}_2^i, \dots, D^i \mathbf{p}_\tau^i)$ and notice that \mathcal{D}^i is fixed. Let $\Delta = \max_{i \in [n]} \|\mathcal{D}^i\|_\infty$. It is clear that $\|\mathcal{D}^i\|_\infty = \Delta \kappa^t$.

To prove Theorem 3, we shall follow the general strategy for proving Theorem 1. We first cut the box $[-\|\mathcal{D}^i\|_\infty, \|\mathcal{D}^i\|_\infty]^s$ into a constant number of small boxes, introduce one mega integer variable for each small box and obtain an MIP, which can be solved in polynomial time. Finally, we round fractional variables to integer variables.

Towards the proof, we first rewrite the input IP in a form that is easier for our analysis.

Step 0: Rewrite the input IP.

We cut the box $[-\|\mathcal{D}^i\|_\infty, \|\mathcal{D}^i\|_\infty]^s$ into $(\frac{2}{\delta})^s$ small δ -boxes of the form $\prod_{i=1}^s [(\lambda_i - 1)\delta\Delta\kappa^t, \lambda_i\delta\Delta\kappa^t]$, where $\lambda_i \in \{-1/\delta + 1, -1/\delta + 2, \dots, 1/\delta\}$, and δ is a sufficiently small constant. In particular, it will become clear later that taking $\delta = \mathcal{O}(\frac{\varepsilon}{st\kappa^t + 1})$ suffices.

We pick a fixed vector within each δ -box and let it be the canonical vector of this δ -box. Each $D^i \mathbf{x}_j^i$ lies in some δ -box and corresponds to the canonical vector of this δ -box. Hence, $\mathcal{D}^i = \mathcal{V}^i + \tilde{\mathcal{D}}^i$ where \mathcal{V}^i is the matrix consisting of the canonical vectors corresponding to the column vectors of \mathcal{D}^i , and $\tilde{\mathcal{D}}^i$ is the “residue matrix” satisfying that $\|\tilde{\mathcal{D}}^i\|_\infty \leq \delta\Delta\kappa^t$.

Since each \mathcal{V}^i has τ columns, it is easy to see that there are $\rho := ((\frac{2}{\delta})^s)^\tau = (\frac{2}{\delta})^{(s\kappa^t)}$ different types of canonical matrices \mathcal{V}^i 's. For $k = 1, \dots, \rho$, let I_k denote the set of indices i 's (of \mathcal{D}^i 's) whose canonical matrix is of type k . The constraint $\sum_{i=1}^n D^i \mathbf{x}^i = \mathbf{b}^0$ essentially asks for selecting some column from each \mathcal{D}^i such that they add up to \mathbf{b}^0 , so we rewrite it with the help of auxiliary variables \mathbf{z} which denote such a choice: Let $z_{ij} = 1$ denote that we select the j -th column of \mathcal{D}^i and $z_{ij} = 0$ otherwise. We have the following equivalent IP:

$$(\text{IP}_3) \quad \min \quad \sum_{i=1}^n \left(\sum_{j=1}^{\tau} \mathbf{w}^i \mathbf{p}_j^i z_{ij} \right) \quad (4a)$$

$$\sum_{k=1}^{\rho} \sum_{i \in I_k} \sum_{j=1}^{\tau} (D^i \mathbf{p}_j^i) z_{ij} = \mathbf{b}^0 \quad (4b)$$

$$\sum_{j=1}^{\tau} z_{ij} = 1, \quad \forall 1 \leq i \leq n \quad (4c)$$

$$z_{ij} \in \{0, 1\}, \quad \forall 1 \leq i \leq n, 1 \leq j \leq \tau \quad (4d)$$

Step 1: Transform (IP_3) to a mixed IP with $\mathcal{O}(1)$ integer variables.

Recall that there are ρ distinct types of canonical matrices \mathcal{V}^i 's. We let $(\mathbf{v}_1^k, \mathbf{v}_2^k, \dots, \mathbf{v}_{\tau}^k)$ be the k -th type. Suppose the canonical matrix of \mathcal{D}^i is of type k , and then $\tilde{\mathcal{D}}^i = (D^i \mathbf{p}_1^i - \mathbf{v}_1^k, D^i \mathbf{p}_2^i - \mathbf{v}_2^k, \dots, D^i \mathbf{p}_{\tau}^i - \mathbf{v}_{\tau}^k)$.

Similar to the proof of Theorem 1, we introduce $y_{kj} = \sum_{i \in I_k} z_{ij}$ as the group variable that counts how many j -th columns we have selected among all \mathcal{D}^i 's whose canonical matrix is of type k . There are $\rho\tau = (\frac{2}{\delta})^{(s\kappa^t)} \kappa^t$ group variables. We establish (MIP_4) below, where y_{kj} 's are integer variables and z_{ij} 's are fractional variables.

$$(\text{MIP}_4) \quad \min \quad \sum_{i=1}^n \left(\sum_{j=1}^{\tau} \mathbf{w}^i \mathbf{p}_j^i z_{ij} \right) \quad (5a)$$

$$\sum_{k=1}^{\rho} \sum_{j=1}^{\tau} \mathbf{v}_j^k y_{kj} + \sum_{k=1}^{\rho} \sum_{i \in I_k} \sum_{j=1}^{\tau} (D^i \mathbf{p}_j^i - \mathbf{v}_j^k) z_{ij} = \mathbf{b}^0 \quad (5b)$$

$$y_{kj} = \sum_{i \in I_k} z_{ij}, \quad \forall 1 \leq j \leq \tau, 1 \leq k \leq \rho \quad (5c)$$

$$\sum_{j=1}^{\tau} z_{ij} = 1, \quad \forall i \in I_k, 1 \leq k \leq \rho \quad (5d)$$

$$z_{ij} \in [0, 1], y_{kj} \in \mathbb{Z}, \quad \forall 1 \leq j \leq \tau, i \in I_k, 1 \leq k \leq \rho \quad (5e)$$

Notice that in (MIP_4) , the coefficients $\mathbf{w}^i \mathbf{p}_j^i$, \mathbf{v}_j^k , and $(D^i \mathbf{p}_j^i - \mathbf{v}_j^k)$ are fixed values instead of variables. Observe that (MIP_4) only contains $\rho\tau$ integer variables, y_{kj} . Thus, applying Kannan's algorithm [40], an optimal solution to (MIP_4) can be computed in $(\rho\tau)^{\mathcal{O}(\rho\tau)} \cdot |I|$ time, which is polynomial.

Let $y_{kj} = y_{kj}^* \in \mathbb{Z}$ and $z_{ij} = z_{ij}^* \in [0, 1]$ be the optimal solution to (MIP_4) .

Step 2: Obtain a feasible solution with $\mathcal{O}(s\tau)$ variables taking a fractional value.

Fix $\mathbf{y} = \mathbf{y}^*$, then (MIP_4) becomes an LP. We see that this LP is a combined system: we denote Constraint (5b) as $C\mathbf{z} = \mathbf{b}_C$, and Constraints (5c) and (5d) as $A_{\Sigma}\mathbf{z} = \mathbf{b}_A$. We denote by LP_{Σ} the linear system obtained by removing $C\mathbf{z} = \mathbf{b}_C$. That is, the constraint matrix of LP_{Σ} is exactly A_{Σ} .

Observe that $A_{\Sigma} = \begin{pmatrix} A_1 & & \\ & \ddots & \\ & & A_{\rho} \end{pmatrix}$ is a block diagonal matrix, as for each $k \in [\rho]$,

the set of variables $I_k \times [\tau]$ appearing in the constraints is disjoint to those for any other $k' \in [\rho] \setminus \{k\}$. Moreover, each A_k is the constraint matrix of an assignment problem and thus totally unimodular (and thus so is A_{Σ}).

Interestingly, for A_k it is easy to find a comparably small kernel element if solutions have enough non-integral components (for integral right-hand sides):

► **Lemma 11.** *For $k \in [\rho]$, consider a vector \mathbf{z} such that $A_k \mathbf{z} = \mathbf{b}$ for some integer vector \mathbf{b} . Let B be the matrix A_k restricted to the columns in $\text{nisupp}(\mathbf{z})$. Then $\text{rank}(B) \leq 2\tau$.*

Proof. If $|\text{nisupp}(\mathbf{z})| \leq 2\tau$, the statement is trivial. Thus, assume $|\text{nisupp}(\mathbf{z})| \geq 2\tau + 1$. For the sake of contradiction, assume $\text{rank}(B) > 2\tau$. Let $I \subseteq \text{nisupp}(\mathbf{z})$ be a set of $|I| = 2\tau + 1$ column indices. Let B' be the matrix B restricted to the columns in I , where afterwards all zero-rows have been removed. Clearly, $2\tau + 1 = \text{rank}(B') \leq \text{rank}(B)$ as we only removed zero-rows. Observe that every row in $\text{rank}(B')$ that corresponds to Constraint (5d) has to contain at least two non-zero variables due to the integral right-hand side. Moreover, all the constraints in (5d) are variable-disjunct. So the number of rows in B' is at most

$$\tau + \frac{|I|}{2} = \tau + \frac{2\tau + 1}{2} < 2\tau + 1 .$$

However, we have $|I| = 2\tau + 1$ columns, so B' has non-zero nullity, which, by Theorem 4, is a contradiction to $\text{rank}(B') = 2\tau + 1$. ◀

► **Lemma 12.** *Let $\hat{\mathbf{z}}$ be a solution to LP_Σ , and B_Σ the matrix A_Σ restricted to the columns $\text{nisupp}(\hat{\mathbf{z}})$. Then we have*

$$\text{nullity}(B_\Sigma) \geq \frac{|\text{nisupp}(\hat{\mathbf{z}})|}{2\tau + 1} .$$

Proof. Observe that since A_Σ is a block diagonal matrix, to prove this lemma, it suffices to show for every $k \in [\rho]$ that B_k , the restriction of A_k to $\text{nisupp}(\hat{\mathbf{z}}) \cap I_k$, we have (where $\eta := |\text{nisupp}(\hat{\mathbf{z}}) \cap I_k|$):

$$\text{nullity}(B_k) \geq \frac{\eta}{2\tau + 1} .$$

If $\eta = 0$, the statement is trivial. If $0 < \eta \leq 2\tau + 1$, we only have to show that the nullity is non-zero, which follows directly from Proposition 10. For $2\tau + 1 < \eta$, by Lemma 11 and rank-nullity theorem, the nullity is at least $\eta - 2\tau$. Observe that for any $\bar{\iota}$ we have

$$\eta - \bar{\iota} \geq \frac{\eta}{2\tau + 1} \iff \eta - \frac{\eta}{2\tau + 1} \geq \bar{\iota} \iff \frac{(2\tau + 1)\eta - \eta}{2\tau + 1} \geq \bar{\iota} \iff \frac{2\tau\eta}{2\tau + 1} \geq \bar{\iota} .$$

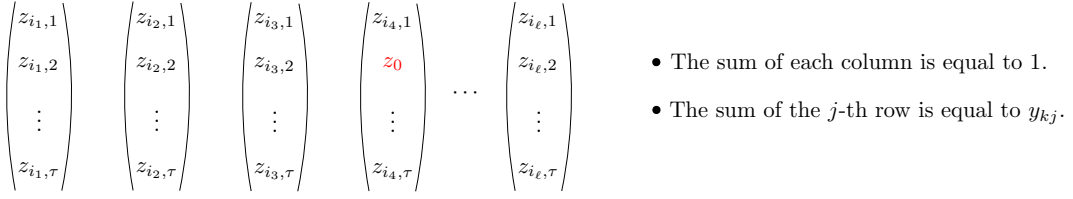
Furthermore, as $\eta > 2\tau + 1$, then $\frac{2\tau\eta}{2\tau + 1} \geq \frac{2\tau(2\tau + 2)}{2\tau + 1} > 2\tau$, i.e., $\eta - \bar{\iota} \geq \frac{\eta}{2\tau + 1}$ holds in particular for $\bar{\iota} := 2\tau$, which eventually leads to

$$\frac{\eta}{2\tau + 1} \leq \eta - 2\tau \leq \text{nullity}(B_k) .$$

Now, let us consider vertex solution γ to the combined linear system, computed by Lemma 7. Let B_Σ be the restriction of A_Σ to the columns in $\text{nisupp}(\gamma)$. We claim that $s \geq \frac{|\text{nisupp}(\gamma)|}{2\tau + 1}$. Assume otherwise. By Lemma 12, $\frac{|\text{nisupp}(\gamma)|}{2\tau + 1} \leq \text{nullity}(B_\Sigma)$. By Lemma 9, having $\text{nullity}(B_\Sigma) > s$ suffices to prove the existence of a non-trivial kernel element on $\text{nisupp}(\gamma)$ of the combined system, that is the one that includes Constraint (5b). This is a contradiction to Lemma 7 which postulates that the constraint matrix restricted to $\text{nisupp}(\gamma)$ is non-singular.

Step 3: Round the $\mathcal{O}(s\tau)$ fractional variables.

We can round the fractional variables for each k , respectively, since the constraints for different k 's cannot affect each other. For a fixed k , by Constraint (5c), we have $y_{kj} =$



■ **Figure 1** Constraints (5c) and (5d) for a fixed k .

$\sum_{i \in I_k} z_{ij}, \forall 1 \leq j \leq \tau$, and by Constraint (5d), it holds that $\sum_{j=1}^{\tau} z_{ij} = 1, \forall i \in I_k$. We assume $I_k = \{i_1, i_2, \dots, i_\ell\}$, and then Constraints (5c) and (5d) are shown as Figure 1.

Check all fractional variables z_{ij} 's, and pick up the fractional variable with the smallest coefficient in the objective function (5a). Without loss of generality, we let z_0 be this fractional variable. See Figure 1 as an illustration. Then we round up the value of z_0 to 1. By the constraint $\sum_{j=1}^{\tau} z_{ij} = 1, \forall i \in I_k$, we round down the values of these variables in the same column as z_0 to zeros. This preserves the constraint $\sum_{j=1}^{\tau} z_{ij} = 1$.

Next, for the fractional variables in the same row as z_0 , we apply a greedy rounding similar as that in Step 3 of Theorem 1. We assume j_0 is the row number that z_0 has. Since $y_{kj} = \sum_{i \in I_k} z_{ij}, \forall 1 \leq j \leq \tau$, without loss of generality we assume that z_{ij_0} 's where $i \in I_k$ in the summation are ordered by the non-decreasing coefficients in the objective function (5a). We compute $\sum_{i \in I_k} (z_{ij_0} - \lfloor z_{ij_0} \rfloor)$. Let $\gamma := \sum_{i \in I_k} (z_{ij_0} - \lfloor z_{ij_0} \rfloor)$. Then we pick out the first γ variables, and round up them to 1 and round down the remaining fractional variables z_{ij_0} 's to zeros. This rounding method will preserve the constraint $y_{kj_0} = \sum_{i \in I_k} z_{ij_0}$.

After the previous two processes, we delete the column and row where z_0 is located. Iteratively find the fractional variable with the smallest coefficient in the objective function from the current fractional variables, and repeat the above operations. Clearly, the constraints $\sum_{j=1}^{\tau} z_{ij} = 1, \forall i \in I_k$ and $y_{kj} = \sum_{i \in I_k} z_{ij}, \forall 1 \leq j \leq \tau$ are always satisfied.

We round up or down all fractional variables by preserving Constraints (5c) and (5d) and meantime not increasing the objective, while this introduces $\mathcal{O}(s\tau\delta\Delta\kappa t)$ error to Constraint (5b). Thus, Theorem 3 follows by taking $\delta = \mathcal{O}(\frac{\varepsilon}{s\tau\kappa t}) = \mathcal{O}(\frac{\varepsilon}{st\kappa^{t+1}})$.

Running time.

$$\text{Step 1: } (\rho\tau)^{\mathcal{O}(\rho\tau)} \cdot |I| = ((\frac{2}{\delta})^{(s\kappa^t)} \kappa^t)^{\mathcal{O}((\frac{2}{\delta})^{(s\kappa^t)} \kappa^t)} \cdot |I| = 2^{(\frac{s\kappa^t}{\varepsilon})^{\mathcal{O}(s\kappa^t)}} \cdot |I|$$

$$\text{Step 2: } (\rho n)^4 \cdot |I| = (\frac{2}{\delta})^{\mathcal{O}(s\kappa^t)} n^4 \cdot |I|$$

$$\text{Step 3: } \rho\tau^2 n^2 \log n = ((\frac{2}{\delta})^s)^{\tau} \tau^2 n^2 \log n = (\frac{2}{\delta})^{\mathcal{O}(s\kappa^t)} n^2 \log n$$

$$\text{All in all, the total time is } 2^{(\frac{s\kappa^t}{\varepsilon})^{\mathcal{O}(s\kappa^t)}} \cdot \text{poly}(|I|), \text{ which has the form of } f(s, t, \kappa, \varepsilon) \text{poly}(|I|).$$

4 Approximation algorithms for n -fold IPs - Proof of Theorem 2

This section is devoted to the proof of Theorem 2. We restate the theorem below.

► **Theorem 2.** *Given is an integer program $\min\{\mathbf{w}\mathbf{x} : \sum_{i=1}^n D^i \mathbf{x}^i = \mathbf{b}^0, A^i \mathbf{x}^i = \mathbf{b}^i, 1 \leq i \leq n, \mathbf{0} \leq \mathbf{x} \leq \mathbf{u}, \mathbf{x} \in \mathbb{Z}^{nt}\}$ with optimal objective value OPT, where $A^i \in \mathbb{Q}_{\geq 0}^{s_A \times t_A}, D^i \in \mathbb{Q}_{\geq 0}^{s_D \times t_D}$, and $t_A = t_D = t$. Let $\varepsilon > 0$ be an arbitrarily small constant. Then there exists an algorithm of running time $f(s_A, s_D, t, \varepsilon) \text{poly}(|I|)$ which returns a near-feasible solution $\tilde{\mathbf{x}}$ such that $\tilde{\mathbf{x}} \in \{\mathbf{x} : (1 - \varepsilon)\mathbf{b}^0 \leq \sum_{i=1}^n D^i \mathbf{x}^i \leq (1 + \varepsilon)\mathbf{b}^0, (1 - \varepsilon)\mathbf{b}^i \leq A^i \mathbf{x}^i \leq (1 + \varepsilon)\mathbf{b}^i, 1 \leq i \leq n, \mathbf{0} \leq \mathbf{x} \leq \mathbf{u}, \mathbf{x} \in \mathbb{Z}^{nt}\}$, and $\mathbf{w}\tilde{\mathbf{x}} \leq \text{OPT}$.*

The above theorem is closely related to Theorem 3. In particular, since all A^i 's are positive matrices, if all entries in A^i is large compared to \mathbf{b}^i , then we know each coordinate of \mathbf{x}^i

must be small. Consequently, \mathbf{x}^i may only take $\mathcal{O}(1)$ possible values, in other words, $\mathbf{x}^i \in \mathcal{P}^i$ where $|\mathcal{P}^i|$ is fixed, and Theorem 3 is applicable. What if some entries of A^i are small, such that some coordinate of \mathbf{x}^i , say, x_j^i , can take a large value? In this case, we observe that when the variable x_j^i increases or decreases by a small amount, the equation $A^i \mathbf{x}^i = \mathbf{b}^i$ may only be violated by some small amount. This motivates us to split \mathbf{x}^i into two parts, the major part and the minor part, where the major part can only take $\mathcal{O}(1)$ possible values, and the minor part can be handled like a fractional variable.

More precisely, for every A_j^i we can multiply it with some λ_j^i to make it big. If some A_j^i is initially big, let its λ_j^i be 1. Then the constraint $A^i \mathbf{x}^i = \mathbf{b}^i$ can be written as

$$A_j^i x_j^i = \lambda_j^i A_j^i \left\lfloor \frac{x_j^i}{\lambda_j^i} \right\rfloor + A_j^i \left(x_j^i - \lambda_j^i \left\lfloor \frac{x_j^i}{\lambda_j^i} \right\rfloor \right),$$

where λ_j^i is the smallest integer such that $\lambda_j^i A_j^i$ is big. Denote $(x_j^i)' := \lfloor \frac{x_j^i}{\lambda_j^i} \rfloor$, and $(x_j^i)'' := x_j^i - \lambda_j^i \lfloor \frac{x_j^i}{\lambda_j^i} \rfloor$. Then we have

$$A_j^i x_j^i = \lambda_j^i A_j^i (x_j^i)' + A_j^i (x_j^i)''.$$

Note that the coefficients of $(x_j^i)'$ are big, while that of $(x_j^i)''$ are not. We can bound $\sum_{j=1}^t A_j^i (x_j^i)''$ by a negligible value due to a simple analysis. It implies that all possible values of $(x_j^i)'$ can be enumerated.

Similarly, the other constraint $\sum_{i=1}^n D^i \mathbf{x}^i = \mathbf{b}^0$ are written as

$$D_j^i x_j^i = \lambda_j^i D_j^i (x_j^i)' + D_j^i (x_j^i)'.$$

Then we formulate a new IP taking $(x_j^i)'$ and $(x_j^i)''$ as variables. In the new IP, the constraint $A^i \mathbf{x}^i = \mathbf{b}^i$ can be safely replaced by $(\mathbf{x}^i)' \in \mathcal{P}^i$. We shall deal with the two summands $\sum_{i=1}^n \sum_{j=1}^t \lambda_j^i D_j^i (x_j^i)'$ and $\sum_{i=1}^n \sum_{j=1}^t D_j^i (x_j^i)''$ separately. The way we deal with the former is the same as Step 1 of Theorem 3, and the way we deal with the latter is the same as Step 1 of Theorem 1. Afterwards, we apply the rounding procedures specified in Theorem 3 and Theorem 1.

We defer the detailed proof of Theorem 2 to Appendix A.

5 Conclusion

We study approximation algorithms for IPs with a constant number of constraints and n -fold IPs. For IPs with a constant number of constraints, our algorithm returns a solution that violates the constraints by an additive error of $\mathcal{O}(\varepsilon \Delta)$. For n -fold IPs, our algorithm returns a solution within a multiplicative error of $1 + \varepsilon$. It is not clear whether similar results can be obtained for other block-structured IPs such as two-stage stochastic IPs or 4-block n -fold IPs. It is also an interesting open problem whether an additive approximation scheme can be obtained for n -fold IPs.

References

- 1 K. Altmanová, D. Knop, and M. Koutecký. Evaluating and tuning n -fold integer programming. *Journal of Experimental Algorithmics*, 24(1):1–22, 2019.
- 2 M. Aschenbrenner and R. Hemmecke. Finiteness theorems in stochastic integer programming. *Foundations of Computational Mathematics*, 7(2):183–227, 2007.

- 3 D. Bertsekas, A. Nedic, and A. Ozdaglar. *Convex analysis and optimization*, volume 1. Athena Scientific, 2003.
- 4 K. Bringmann and A. Cassis. Faster knapsack algorithms via bounded monotone min-plus-convolution. In *Proceedings of the 49th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 229 of *LIPIcs*, pages 31:1–31:21, 2022.
- 5 M. Buchem, L. Rohwedder, T. Vredeveld, and A. Wiese. Additive approximation schemes for load balancing problems. In *Proceedings of the 48th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 198 of *LIPIcs*, pages 42:1–42:17, 2021.
- 6 T. M. Chan. Approximation schemes for 0-1 knapsack. In *1st Symposium on Simplicity in Algorithms (SOSA)*, pages 5–1, 2018.
- 7 H. Chen, L. Chen, and G. Zhang. Block-structured integer programming: Can we parameterize without the largest coefficient? *Discrete Optimization*, 46:100743, 2022.
- 8 H. Chen, L. Chen, and G. Zhang. FPT algorithms for a special block-structured integer program with applications in scheduling. *Mathematical Programming*, 208:463–496, 2024.
- 9 L. Chen, M. Koutecký, L. Xu, and W. Shi. New bounds on augmenting steps of block-structured integer programs. In *Proceedings of the 28th Annual European Symposium on Algorithms (ESA)*, volume 173 of *LIPIcs*, pages 33:1–33:19, 2020.
- 10 L. Chen, J. Lian, Y. Mao, and G. Zhang. Approximating partition in near-linear time. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing (STOC)*, pages 307–318, 2024.
- 11 L. Chen, J. Lian, Y. Mao, and G. Zhang. A nearly quadratic-time fptas for knapsack. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing (STOC)*, pages 283–294, 2024.
- 12 J. Cslovjceksek, F. Eisenbrand, C. Hunkenschroder, L. Rohwedder, and R. Weismantel. Block-structured integer and linear programming in strongly polynomial and near linear time. In *Proceedings of the 32nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1666–1681. SIAM, 2021.
- 13 J. Cslovjceksek, F. Eisenbrand, M. Pilipczuk, M. Venzin, and R. Weismantel. Efficient sequential and parallel algorithms for multistage stochastic integer programming using proximity. In *Proceedings of the 29th Annual European Symposium on Algorithms (ESA)*, volume 204 of *LIPIcs*, pages 33:1–33:14, 2021.
- 14 J. Cslovjceksek, M. Koutecký, A. Lassota, M. Pilipczuk, and A. Polak. Parameterized algorithms for block-structured integer programs with large entries. In *Proceedings of the 35th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 740–751. SIAM, 2024.
- 15 D. Dadush. *Integer programming, lattice algorithms, and deterministic volume estimation*. Georgia Institute of Technology, 2012.
- 16 D. Dadush. A randomized sieving algorithm for approximate integer programming. *Algorithmica*, 70(2):208–244, 2014.
- 17 D. Dadush, F. Eisenbrand, and T. Rothvoss. From approximate to exact integer programming. *Mathematical Programming*, pages 1–19, 2024.
- 18 D. Dadush, C. Peikert, and S. Vempala. Enumerative lattice algorithms in any norm via M-ellipsoid coverings. In *Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 580–589. IEEE, 2011.
- 19 J. A. De Loera, R. Hemmecke, S. Onn, and R. Weismantel. N-fold integer programming. *Discrete Optimization*, 5(2):231–241, 2008.

- 20 M. Deng, C. Jin, and X. Mao. Approximating knapsack and partition via dense subset sums. In *Proceedings of the 34th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2961–2979. SIAM, 2023.
- 21 S. Dye, L. Stougie, and A. Tomasgard. The stochastic single resource service-provision problem. *Naval Research Logistics*, 50(8):869–887, 2003.
- 22 F. Eisenbrand, C. Hunkenschroder, and K. M. Klein. Faster algorithms for integer programs with block structure. In *Proceedings of the 45th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 107 of *LIPIcs*, pages 49:1–49:13, 2018.
- 23 F. Eisenbrand, C. Hunkenschroder, K. M. Klein, M. Koutecký, A. Levin, and S. Onn. An algorithmic theory of integer programming. *arXiv preprint arXiv:1904.01361*, 2019.
- 24 F. Eisenbrand and R. Weismantel. Proximity results and faster algorithms for integer programming using the Steinitz Lemma. *ACM Transactions on Algorithms*, 16(1):1–14, 2019.
- 25 A. V. Fishkin, O. Gerber, K. Jansen, and R. Solis-Oba. On packing squares with resource augmentation: Maximizing the profit. In *Proceedings of the 2005 Australasian Symposium on Theory of Computing-Volume 41*, pages 61–67, 2005.
- 26 A. Gupta, M. Pál, R. Ravi, and A. Sinha. Boosted sampling: Approximation algorithms for stochastic optimization. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC)*, pages 417–426, 2004.
- 27 R. Hemmecke, M. Köppe, and R. Weismantel. Graver basis and proximity techniques for block-structured separable convex integer minimization problems. *Mathematical Programming*, 145(1-2):1–18, 2014.
- 28 R. Hemmecke, S. Onn, and L. Romanchuk. N-fold integer programming in cubic time. *Mathematical Programming*, 137(1-2):325–341, 2013.
- 29 R. Hemmecke and R. Schultz. Decomposition of test sets in stochastic integer programming. *Mathematical Programming*, 94(2-3):323–341, 2003.
- 30 O. H. Ibarra and C. E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM*, 22(4):463–468, 1975.
- 31 N. Immorlica, D. Karger, M. Minkoff, and V. S. Mirrokni. On the costs and benefits of procrastination: Approximation algorithms for stochastic combinatorial optimization problems. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 691–700, 2004.
- 32 K. Iwama and G. Zhang. Online knapsack with resource augmentation. *Information Processing Letters*, 110(22):1016–1020, 2010.
- 33 K. Jansen, K. M. Klein, and A. Lassota. The double exponential runtime is tight for 2-stage stochastic ILPs. In *Proceedings of the 22nd International Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 297–310. Springer, 2021.
- 34 K. Jansen, K. M. Klein, M. Maack, and M. Rau. Empowering the configuration-IP: New PTAS results for scheduling with setup times. *Mathematical Programming*, 195(1-2):367–401, 2022.
- 35 K. Jansen, A. Lassota, and L. Rohwedder. Near-linear time algorithm for n-fold ILPs via color coding. *SIAM Journal on Discrete Mathematics*, 34(4):2282–2299, 2020.
- 36 K. Jansen and M. Mastrolilli. Scheduling unrelated parallel machines: Linear programming strikes back. university of kiel. Technical report, Technical Report 1004, 2010.
- 37 K. Jansen and L. Rohwedder. On integer programming, discrepancy, and convolution. *Mathematics of Operations Research*, 48(3):1481–1495, 2023.

- 38 C. Jin. An improved FPTAS for 0-1 knapsack. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 132 of *LIPIcs*, pages 76:1–76:14, 2019.
- 39 H. W. Lenstra Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, 1983.
- 40 R. Kannan. Minkowski’s convex body theorem and integer programming. *Mathematics of Operations Research*, 12(3):415–440, 1987.
- 41 R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Springer, 1972.
- 42 K. M. Klein. About the complexity of two-stage stochastic IPs. *Mathematical Programming*, pages 1–19, 2021.
- 43 D. Knop and M. Koutecký. Scheduling meets n-fold integer programming. *Journal of Scheduling*, 21(5):493–503, 2018.
- 44 D. Knop, M. Koutecký, and M. Mnich. Combinatorial n-fold integer programming and applications. *Mathematical Programming*, 184(1):1–34, 2020.
- 45 D. Knop, M. Pilipczuk, and M. Wrochna. Tight complexity lower bounds for integer linear programming with few constraints. *ACM Transactions on Computation Theory*, 12(3):1–19, 2020.
- 46 M. Koutecký, A. Levin, and S. Onn. A parameterized strongly polynomial algorithm for block structured integer programs. In *Proceedings of the 45th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 107 of *LIPIcs*, pages 85:1–85:14, 2018.
- 47 X. Mao. $(1 - \varepsilon)$ -approximation of knapsack in nearly quadratic time. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing (STOC)*, pages 295–306, 2024.
- 48 M. Mucha, K. Węgrzycki, and M. Włodarczyk. A subquadratic approximation scheme for partition. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 70–88. SIAM, 2019.
- 49 T. Oertel, J. Paat, and R. Weismantel. A colorful Steinitz Lemma with application to block-structured integer programs. *Mathematical Programming*, 204(1):677–702, 2024.
- 50 C. H. Papadimitriou. On the complexity of integer programming. *Journal of the ACM*, 28(4):765–768, 1981.
- 51 R. Ravi and A. Sinha. Hedging uncertainty: Approximation algorithms for stochastic optimization problems. *Mathematical Programming*, 108:97–114, 2006.
- 52 V. Reis and T. Rothvoss. The subspace flatness conjecture and faster integer programming. In *Proceedings of the 64th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 974–988. IEEE, 2023.
- 53 D. B. Shmoys and C. Swamy. Stochastic optimization is (almost) as easy as deterministic optimization. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 228–237. IEEE, 2004.
- 54 C. Swamy and D. B. Shmoys. Sampling-based approximation algorithms for multi-stage stochastic optimization. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 357–366. IEEE, 2005.

A Proof of Theorem 2

► **Theorem 2.** *Given is an integer program $\min\{\mathbf{w}\mathbf{x} : \sum_{i=1}^n D^i \mathbf{x}^i = \mathbf{b}^0, A^i \mathbf{x}^i = \mathbf{b}^i, 1 \leq i \leq n, \mathbf{0} \leq \mathbf{x} \leq \mathbf{u}, \mathbf{x} \in \mathbb{Z}^{nt}\}$ with optimal objective value OPT , where $A^i \in \mathbb{Q}_{\geq 0}^{s_A \times t_A}$, $D^i \in \mathbb{Q}_{\geq 0}^{s_D \times t_D}$, and $t_A = t_D = t$. Let $\varepsilon > 0$ be an arbitrarily small constant. Then there exists an algorithm of running time $f(s_A, s_D, t, \varepsilon) \text{poly}(|I|)$ which returns a near-feasible solution $\tilde{\mathbf{x}}$ such that $\tilde{\mathbf{x}} \in \{\mathbf{x} : (1 - \varepsilon)\mathbf{b}^0 \leq \sum_{i=1}^n D^i \mathbf{x}^i \leq (1 + \varepsilon)\mathbf{b}^0, (1 - \varepsilon)\mathbf{b}^i \leq A^i \mathbf{x}^i \leq (1 + \varepsilon)\mathbf{b}^i, 1 \leq i \leq n, \mathbf{0} \leq \mathbf{x} \leq \mathbf{u}, \mathbf{x} \in \mathbb{Z}^{nt}\}$, and $\mathbf{w}\tilde{\mathbf{x}} \leq \text{OPT}$.*

Proof. Since $A^i \mathbf{x}^i = \mathbf{b}^i$, $\mathbf{x} \geq \mathbf{0}$, and $A^i \in \mathbb{Q}_{\geq 0}^{s_A \times t_A}$, we know $\mathbf{b}^i \in \mathbb{Q}_{\geq 0}^{s_A}$. For ease of discussion, we scale the values of all non-zero components of each vector \mathbf{b}^i to 1, and keep the values on the zero components unchanged. By doing so, we have that $\mathbf{b}^i \in \{0, 1\}^{s_A}$ for all $i = 1, \dots, n$.

Let A_j^i be the j -th column of matrix A^i . By the constraint $A^i \mathbf{x}^i = \mathbf{b}^i$, we have

$$A_1^i x_1^i + A_2^i x_2^i + \dots + A_t^i x_t^i = \mathbf{b}^i. \quad (6)$$

If, say, some h -th coordinate of \mathbf{b}^i is 0, we consider the h -th coordinate of A_j^i 's. There are two possibilities: (i). The h -th coordinate of some A_j^i is nonzero. Then $x_j^i = 0$ in any feasible solution. We simply fix $x_j^i = 0$ and ignore this variable (as well as its corresponding A_j^i) throughout our following discussion. (ii). Assume without loss of generality that the h -th coordinate of every A_j^i is 0. Then the constraint induced by the h -th coordinates becomes $\sum_{j=1}^t 0 \cdot x_j^i = 0$, and we simply remove this constraint, i.e., we remove the h -th coordinate from \mathbf{b}^i and A_j^i . Hence, without loss of generality we may assume that $\mathbf{b}^i = \mathbf{1}$ for all i . From now on we still write \mathbf{b}^i for consistency, but notice that during all our subsequent discussion its zero entries are ignored.

Step 0: Classify A_j^i 's as big or small, and establish an equivalent IP.

We fix $\psi := \frac{\varepsilon}{2t}$ and classify all A_j^i 's into two different groups: If there is a component in vector A_j^i with a value at least ψ , then the vector A_j^i is defined as **big**. Otherwise, if the values on all components of vector A_j^i are less than ψ , then the vector A_j^i is defined as **small**. According to Eq(6), there are at most $\mathcal{O}(1/\psi)$ big A_j^i 's in every dimension. Since $\mathbf{b}^i = \mathbf{1}$, there are at most $\mathcal{O}(s_A/\psi)$ big A_j^i 's for each fixed i , that is, there are at most $\mathcal{O}(s_A/\psi)$ non-zero x_j^i 's whose corresponding coefficients are big vectors for each fixed i .

If all A_j^i 's are big vectors in Eq(6) for all i 's, then we get that $\sum_{j=1}^t x_j^i = \mathcal{O}(s_A/\psi)$, which implies that $\|\mathbf{x}^i\|_\infty \leq \|\mathbf{x}^i\|_1 = \mathcal{O}(s_A/\psi)$. By Theorem 3, there exists an algorithm of running time $2^{2^{\mathcal{O}(s_D(s_A/\varepsilon)t^{\mathcal{O}(1)})}} \cdot \text{poly}(|I|) = f(s_A, s_D, t, \varepsilon) \text{poly}(|I|)$ which returns a near-feasible solution $\tilde{\mathbf{x}}$ such that $\tilde{\mathbf{x}} \in \{\mathbf{x} : (1 - \varepsilon)\mathbf{b}^0 \leq \sum_{i=1}^n D^i \mathbf{x}^i \leq (1 + \varepsilon)\mathbf{b}^0, A^i \mathbf{x}^i = \mathbf{b}^i, \mathbf{0} \leq \mathbf{x} \leq \mathbf{u}, \mathbf{x} \in \mathbb{Z}^{nt}\}$, and $\mathbf{w}\tilde{\mathbf{x}} \leq \text{OPT}$. Theorem 2 follows.

Hence, from now on we assume there exists some small vector A_j^i in Eq(6) for some i . In this case, for every A_j^i we multiply it with some λ_j^i to make it big. More precisely, Eq(6) can be written as

$$A_j^i x_j^i = \lambda_j^i A_j^i \left\lfloor \frac{x_j^i}{\lambda_j^i} \right\rfloor + A_j^i \left(x_j^i - \lambda_j^i \left\lfloor \frac{x_j^i}{\lambda_j^i} \right\rfloor \right), \quad (7)$$

where the value of $\lambda_j^i \geq 1, \lambda_j^i \in \mathbb{Z}$ is chosen to make the values of some components of $\lambda_j^i A_j^i$ no less than ψ , i.e., λ_j^i is the smallest integer such that $\lambda_j^i A_j^i$ is big.

Let $(x_j^i)' := \lfloor \frac{x_j^i}{\lambda_j^i} \rfloor$, and $(x_j^i)'' := x_j^i - \lambda_j^i \lfloor \frac{x_j^i}{\lambda_j^i} \rfloor$. Then we have

$$A_j^i x_j^i = \lambda_j^i A_j^i (x_j^i)' + A_j^i (x_j^i)''. \quad (8)$$

We have the following straightforward observation.

► **Observation 13.** $(x_j^i)'' < \lambda_j^i$;

- $(x_j^i)'' = 0$ if $\lambda_j^i = 1$;
- If $(x_j^i)'' > 0$, then all coordinates of A_j^i are less than ψ .
- If $(x_j^i)'' > 0$, then $\|\lambda_j^i A_j^i\|_\infty \leq 2\psi$.

Similarly, let D_j^i be the j -th column of matrix D^i . The constraint $\sum_{i=1}^n D^i \mathbf{x}^i = \mathbf{b}^0$ can be rewritten as

$$\sum_{i=1}^n \sum_{j=1}^t D_j^i x_j^i = \mathbf{b}^0, \quad (9)$$

where $D_j^i x_j^i$ can be further rewritten as

$$D_j^i x_j^i = \lambda_j^i D_j^i (x_j^i)' + D_j^i (x_j^i)''.$$

Note that the two vectors $\lambda_j^i D_j^i$ and D_j^i are not necessarily big (or small). Now we can rewrite the input IP as follows.

$$(\text{IP}_5) \quad \min \quad \sum_{i=1}^n \sum_{j=1}^t w_j^i ((x_j^i)' + (x_j^i)'') \quad (10a)$$

$$\sum_{i=1}^n \sum_{j=1}^t (\lambda_j^i D_j^i (x_j^i)' + D_j^i (x_j^i)'') = \mathbf{b}^0 \quad (10b)$$

$$\sum_{j=1}^t (\lambda_j^i A_j^i (x_j^i)' + A_j^i (x_j^i)'') = \mathbf{b}^i, \quad \forall 1 \leq i \leq n \quad (10c)$$

$$(x_j^i)' \in [0, (u_j^i)'], (x_j^i)'' \in [0, (u_j^i)''], \quad \forall 1 \leq i \leq n, 1 \leq j \leq t \quad (10d)$$

$$(x_j^i)' \in \mathbb{Z}, (x_j^i)'' \in \mathbb{Z}, \quad \forall 1 \leq i \leq n, 1 \leq j \leq t \quad (10e)$$

Recall that $0 \leq x_j^i \leq u_j^i$, so here $(u_j^i)' := \lfloor \frac{u_j^i}{\lambda_j^i} \rfloor$ and $(u_j^i)'' := u_j^i - \lambda_j^i \lfloor \frac{u_j^i}{\lambda_j^i} \rfloor$.

Step 1: Transform (IP₅) to a mixed IP with $\mathcal{O}(1)$ integer variables.

Towards the transformation, we first replace the constraint $A^i \mathbf{x}^i = \mathbf{b}^i$ with something similar to $\mathbf{x}^i \in \mathcal{P}^i$.

We first consider the “minor part”, $(x_j^i)''$. By Observation 13, we have that

$$\sum_{j=1}^t A_j^i (x_j^i)'' = \sum_{j: \lambda_j^i \neq 1} A_j^i (x_j^i)'' + \sum_{j: \lambda_j^i = 1} A_j^i (x_j^i)'' = \sum_{j: \lambda_j^i \neq 1} A_j^i (x_j^i)'.$$

Since $(x_j^i)'' < \lambda_j^i$, it always holds that

$$\sum_{j: \lambda_j^i \neq 1} A_j^i (x_j^i)'' < \sum_{j: \lambda_j^i \neq 1} A_j^i \lambda_j^i \leq t \cdot 2(\psi, \psi, \dots, \psi)^\top = 2t(\frac{\varepsilon}{2t}, \frac{\varepsilon}{2t}, \dots, \frac{\varepsilon}{2t})^\top = (\varepsilon, \varepsilon, \dots, \varepsilon)^\top.$$

This means, whatever the value of $(x_j^i)''$ is, its contribution to $A^i \mathbf{x}^i$ is negligible.

Next, we consider the “major part”, $(x_j^i)'$. The fact that $\mathbf{b}^i \in \{0, 1\}^{s_A}$ and $\lambda_j^i A_j^i$ is big guarantees that $(\mathbf{x}^i)'$ can only take a few distinct values. More precisely, it is easy to verify that $\|(\mathbf{x}^i)'\|_1 = \mathcal{O}(s_A/\psi)$, and hence we can obtain all possible feasible solutions to $(1 - \varepsilon)\mathbf{b}^i \leq A^i \mathbf{x}^i \leq (1 + \varepsilon)\mathbf{b}^i$ by a straightforward enumeration in $\mathcal{O}(s_A/\psi)^t = \mathcal{O}(s_A t/\varepsilon)^t$ time. Denote by \mathcal{P}^i the set of these solutions.

Now it is easy to see that the constraint $A^i \mathbf{x}^i = \mathbf{b}^i$ can be safely replaced by $(\mathbf{x}^i)' \in \mathcal{P}^i$.

Next, we deal with $\sum_{i=1}^n \sum_{j=1}^t (\lambda_j^i D_j^i (x_j^i)' + D_j^i (x_j^i)'') = \mathbf{b}^0$. We shall deal with the two summands separately. The way we deal with $\sum_{i=1}^n \sum_{j=1}^t \lambda_j^i D_j^i (x_j^i)'$ is the same as Step

1 of Theorem 3, and the way we deal with $\sum_{i=1}^n \sum_{j=1}^t D_j^i(x_j^i)''$ is the same as Step 1 of Theorem 1.

Step 1.1: Deal with $\sum_{i=1}^n \sum_{j=1}^t \lambda_j^i D_j^i(x_j^i)'$:

Let $\kappa := \|(\mathbf{x}^i)'\|_\infty = \mathcal{O}(s_A/\psi)$. Recall that $(\mathbf{x}^i)' \in \mathcal{P}^i$ and let $\tau := \max_{i \in [n]} |\mathcal{P}^i| = \kappa^t$. By adding dummy vectors, we may assume $|\mathcal{P}^i| = \tau$ for all i . Let $\mathcal{P}^i = \{(\mathbf{x}_1^i)', (\mathbf{x}_2^i)', \dots, (\mathbf{x}_\tau^i)'\}$. Note that from now on every $(\mathbf{x}_j^i)'$ is a fixed value rather than a variable. Define $\mathcal{D}^i = ((D^i)'(\mathbf{x}_1^i)', (D^i)'(\mathbf{x}_2^i)', \dots, (D^i)'(\mathbf{x}_\tau^i)'),$ where $(D^i)' = (\lambda_1^i D_1^i, \lambda_2^i D_2^i, \dots, \lambda_t^i D_t^i)$. The following is essentially the same as Step 1 of Theorem 3 except that the parameters may take different values. For the completeness of this paper, we present all the details.

Let $\Delta_1 = \max_{i \in [n]} \|(D^i)'\|_\infty$. It is clear that $\|\mathcal{D}^i\|_\infty = \mathcal{O}(s_A/\phi)\Delta_1 = \mathcal{O}(s_A t \Delta_1/\varepsilon)$. We cut the box $[-\|\mathcal{D}^i\|_\infty, \|\mathcal{D}^i\|_\infty]^{s_D}$ into $(\frac{2}{\delta_1})^{s_D}$ small δ_1 -boxes of the form $\prod_{i=1}^{s_D} [(\lambda_i - 1)\delta_1 \|\mathcal{D}^i\|_\infty, \lambda_i \delta_1 \|\mathcal{D}^i\|_\infty]$, where we shall pick a sufficiently small $\delta_1 = \frac{\varepsilon}{s_D t \kappa^t + 1} = \frac{1}{s_D} (\frac{\varepsilon}{s_A t})^{\mathcal{O}(t)}$. All the $(D^i)'(\mathbf{x}_\phi^i)'$'s that are in the same small box are called similar.

We pick a fixed vector within each δ_1 -box and let it be the canonical vector of this δ_1 -box. Each $(D^i)'(\mathbf{x}_\phi^i)'$ lies in some δ_1 -box, and we say it corresponds to the canonical vector of this δ_1 -box. Hence, we may write $\mathcal{D}^i = \mathcal{V}^i + \tilde{\mathcal{D}}^i$ where \mathcal{V}^i is the matrix consisting of the canonical vectors corresponding to the column vectors of \mathcal{D}^i , and $\tilde{\mathcal{D}}^i$ is the residue matrix satisfying that $\|\tilde{\mathcal{D}}^i\|_\infty \leq \delta_1 \|\mathcal{D}^i\|_\infty = \mathcal{O}(\delta_1 s_A t \Delta_1/\varepsilon)$. Furthermore, we make two observations:

(i). There are $\rho := ((\frac{2}{\delta_1})^{s_D})^\tau = (\frac{2}{\delta_1})^{s_D \tau}$ different types of canonical matrix \mathcal{V}^i 's since each \mathcal{V}^i has τ columns. For $k = 1, \dots, \rho$, let I_k denote the set of indices of \mathcal{D}^i 's whose canonical matrix is of type k . Denote by $(\mathbf{v}_1^k, \mathbf{v}_2^k, \dots, \mathbf{v}_\tau^k)$ the canonical matrix of type k . Then if the canonical matrix of \mathcal{D}^i is of type k , we have that

$$\tilde{\mathcal{D}}^i = \mathcal{D}^i - \mathcal{V}^i = ((D^i)'(\mathbf{x}_1^i)' - \mathbf{v}_1^k, (D^i)'(\mathbf{x}_2^i)' - \mathbf{v}_2^k, \dots, (D^i)'(\mathbf{x}_\tau^i)' - \mathbf{v}_\tau^k).$$

(ii). With $(\mathbf{x}^i)' \in \mathcal{P}^i$, the constraint $\sum_{i=1}^n (D^i)'(\mathbf{x}^i)'$ can be interpreted as selecting some column from each \mathcal{D}^i . Hence, we introduce a binary variable $z_{i\phi}$ such that $z_{i\phi} = 1$ denotes that we select the ϕ -th column of \mathcal{D}^i , and $z_{i\phi} = 0$ otherwise. Notice that for $i \in I_k$, \mathcal{D}^i corresponds to the canonical matrix of type k . We define $y_{k\phi} := \sum_{i \in I_k} z_{i\phi}$, where $y_{k\phi}$ can be interpreted as counting how many times the ϕ -th column of the type- k canonical matrix are selected. With the new variables, we may rewrite the constraint as:

$$\sum_{i=1}^n \sum_{j=1}^t \lambda_j^i D_j^i(x_j^i)' = \sum_{k=1}^{\rho} \sum_{\phi=1}^{\tau} \mathbf{v}_\phi^k y_{k\phi} + \sum_{k=1}^{\rho} \sum_{i \in I_k} \sum_{\phi=1}^{\tau} ((D^i)'(\mathbf{x}_\phi^i)' - \mathbf{v}_\phi^k) z_{i\phi}, \quad (11)$$

together with two extra constraints:

$$y_{k\phi} = \sum_{i \in I_k} z_{i\phi}, \quad \forall 1 \leq \phi \leq \tau, 1 \leq k \leq \rho, \quad (12)$$

and

$$\sum_{\phi=1}^{\tau} z_{i\phi} = 1, \quad \forall i \in I_k, 1 \leq k \leq \rho. \quad (13)$$

Following this notation, $\sum_{i=1}^n \sum_{j=1}^t w_j^i(x_j^i)'$ in (10a) can be rewritten as $\sum_{i=1}^n (\sum_{\phi=1}^{\tau} \mathbf{w}^i(\mathbf{x}_\phi^i)' z_{i\phi})$.

Step 1.2: Deal with $\sum_{i=1}^n \sum_{j=1}^t D_j^i(x_j^i)''$:

This part is essentially the same as Step 1 of Theorem 1, except that the parameters may take different values. We present the details for completeness.

Let $\Delta_2 = \max_{i \in [n]} \|D_j^i\|_\infty$, then $D_j^i \in [-\Delta_2, \Delta_2]^{s_D}$. We cut $[-\Delta_2, \Delta_2]^{s_D}$ into $(2/\delta_2)^{s_D}$ small boxes (called δ_2 -boxes), where $\delta_2 = \mathcal{O}(\frac{\varepsilon}{s_D})$. Each δ_2 -box is of the form $\prod_{i=1}^{s_D} [(\lambda_i -$

$1)\delta_2\Delta_2, \lambda_i\delta_2\Delta_2]$, where $\lambda_i \in \{-1/\delta_2 + 1, -1/\delta_2 + 2, \dots, 1/\delta_2\}$. We index all the δ_2 -boxes arbitrarily, and let I_d be the set of indices (i, j) 's of D_j^i 's that are in the d -th δ_2 -box, where $d = 1, \dots, (2/\delta_2)^{s_D}$.

For the d -th δ_2 -box, we pick arbitrary one fixed vector \mathbf{v}_d as its canonical vector. For each $(i, j) \in I_d$, we define $\tilde{D}_j^i = D_j^i - \mathbf{v}_d$, and then it is clear that $\|\tilde{D}_j^i\|_\infty \leq \delta_2\Delta_2$. Now we can rewrite $\sum_{i=1}^n \sum_{j=1}^t D_j^i(x_j^i)''$ as follows by introducing a new variable

$$y_d := \sum_{(i,j) \in I_d} (x_j^i)'', \quad \forall 1 \leq d \leq (2/\delta_2)^{s_D}, \quad (14)$$

and thus

$$\sum_{i=1}^n \sum_{j=1}^t D_j^i(x_j^i)'' = \sum_{d=1}^{(2/\delta_2)^{s_D}} \sum_{(i,j) \in I_d} (\mathbf{v}_d + \tilde{D}_j^i)(x_j^i)'' = \sum_{d=1}^{(2/\delta_2)^{s_D}} \mathbf{v}_d y_d + \sum_{d=1}^{(2/\delta_2)^{s_D}} \sum_{(i,j) \in I_d} \tilde{D}_j^i(x_j^i)'', \quad (15)$$

Now we are ready to reformulate (IP₅) as a mixed IP. Combining the equations (11), (12), (13), (14), (15) with the objective function, we have

$$(\text{MIP}_6) \quad \min \quad \sum_{i=1}^n \left(\sum_{\phi=1}^{\tau} \mathbf{w}^i(\mathbf{x}_\phi^i)' z_{i\phi} \right) + \sum_{i=1}^n \sum_{j=1}^t w_j^i(x_j^i)'' \quad (16a)$$

$$\begin{aligned} & \sum_{k=1}^{\rho} \sum_{\phi=1}^{\tau} \mathbf{v}_\phi^k y_{k\phi} + \sum_{k=1}^{\rho} \sum_{i \in I_k} \sum_{\phi=1}^{\tau} ((D^i)'(\mathbf{x}_\phi^i)' - \mathbf{v}_\phi^k) z_{i\phi} \\ & + \sum_{d=1}^{(2/\delta_2)^{s_D}} \mathbf{v}_d y_d + \sum_{d=1}^{(2/\delta_2)^{s_D}} \sum_{(i,j) \in I_d} \tilde{D}_j^i(x_j^i)'' = \mathbf{b}^0 \end{aligned} \quad (16b)$$

$$y_{k\phi} = \sum_{i \in I_k} z_{i\phi}, \quad \forall 1 \leq \phi \leq \tau, 1 \leq k \leq \rho \quad (16c)$$

$$\sum_{\phi=1}^{\tau} z_{i\phi} = 1, \quad \forall i \in I_k, 1 \leq k \leq \rho \quad (16d)$$

$$\sum_{(i,j) \in I_d} (x_j^i)'' = y_d, \quad \forall 1 \leq d \leq (2/\delta_2)^{s_D} \quad (16e)$$

$$y_{k\phi} \in \mathbb{Z}, z_{i\phi} \in [0, 1], \quad \forall 1 \leq \phi \leq \tau, i \in I_k, 1 \leq k \leq \rho \quad (16f)$$

$$y_d \in \mathbb{Z}, (x_j^i)'' \in [0, (u_j^i)''], \quad \forall (i, j) \in I_d, 1 \leq d \leq (2/\delta_2)^{s_D} \quad (16g)$$

Observe that (MIP₆) relaxes $z_{i\phi} \in \{0, 1\}$ to $z_{i\phi} \in [0, 1]$ and removes the constraint $(x_j^i)'' \in \mathbb{Z}$. Thus, it only contains $\rho\tau$ integer variables, $y_{k\phi}$, and $(2/\delta_2)^{s_D}$ integer variables, y_d . Thus, applying Kannan's algorithm [40], an optimal solution to (MIP₆) can be computed in $(\rho\tau + (2/\delta_2)^{s_D})^{\mathcal{O}(\rho\tau + (2/\delta_2)^{s_D})} \cdot |I|$ time. Recall that $\rho = (\frac{2}{\delta_1})^{s_D\tau}$, $\tau = \mathcal{O}(s_A t/\varepsilon)^t$, $\delta_1 = \frac{1}{s_D}(\frac{\varepsilon}{s_A t})^{\mathcal{O}(t)}$, and $\delta_2 = \mathcal{O}(\frac{\varepsilon}{s_D})$, so (MIP₆) can be computed in $2^{(2^{(s_D \cdot (s_A t/\varepsilon)^t)^{\mathcal{O}(1)} + (s_D/\varepsilon)^{\mathcal{O}(s_D)})^{\mathcal{O}(1)}})} \cdot |I| = f(s_A, s_D, t, \varepsilon) \cdot |I|$ time.

Let $y_{k\phi} = y_{k\phi}^*$, $y_d = y_d^*$, $(x_j^i)'' = (x_j^i)''^*$, and $z_{i\phi} = z_{i\phi}^*$ be the optimal solution to (MIP₆). Note that $y_{k\phi}^* \in \mathbb{Z}$, $y_d^* \in \mathbb{Z}$. By fixing those integer variables, (MIP₆) becomes an LP with its optimal solution being $(x_j^i)''^* \in \mathbb{R}$, and $z_{i\phi}^* \in [0, 1]$. The following step is devoted to rounding these variables.

Step 2: Obtain a feasible solution with $\mathcal{O}(s_D\tau)$ variables taking a fractional value.

When we round the fractional variables, we can decompose (MIP₆) into the following two separate LPs:

$$(\text{LP}_7) \quad \min \quad \sum_{i=1}^n \left(\sum_{\phi=1}^{\tau} \mathbf{w}^i(\mathbf{x}_{\phi}^i)' z_{i\phi} \right) \quad (17a)$$

$$\begin{aligned} & \sum_{k=1}^{\rho} \sum_{\phi=1}^{\tau} \mathbf{v}_{\phi}^k y_{k\phi} + \sum_{k=1}^{\rho} \sum_{i \in I_k} \sum_{\phi=1}^{\tau} ((D^i)'(\mathbf{x}_{\phi}^i)' - \mathbf{v}_{\phi}^k) z_{i\phi} \\ &= \sum_{k=1}^{\rho} \sum_{\phi=1}^{\tau} \mathbf{v}_{\phi}^k y_{k\phi}^* + \sum_{k=1}^{\rho} \sum_{i \in I_k} \sum_{\phi=1}^{\tau} ((D^i)'(\mathbf{x}_{\phi}^i)' - \mathbf{v}_{\phi}^k) z_{i\phi}^* \end{aligned} \quad (17b)$$

$$\sum_{i \in I_k} z_{i\phi} = y_{k\phi}^*, \quad \forall 1 \leq \phi \leq \tau, 1 \leq k \leq \rho \quad (17c)$$

$$\sum_{\phi=1}^{\tau} z_{i\phi} = 1, \quad \forall i \in I_k, 1 \leq k \leq \rho \quad (17d)$$

$$z_{i\phi} \in [0, 1], \quad \forall 1 \leq \phi \leq \tau, i \in I_k, 1 \leq k \leq \rho \quad (17e)$$

$$(\text{LP}_8) \quad \min \quad \sum_{i=1}^n \sum_{j=1}^t w_j^i (x_j^i)'' \quad (18a)$$

$$\sum_{d=1}^{(2/\delta_2)^{s_D}} \sum_{(i,j) \in I_d} \tilde{D}_j^i (x_j^i)'' = \sum_{d=1}^{(2/\delta_2)^{s_D}} \sum_{(i,j) \in I_d} \tilde{D}_j^i (x_j^i)''^* \quad (18b)$$

$$\sum_{(i,j) \in I_d} (x_j^i)'' = y_d^*, \quad \forall 1 \leq d \leq (2/\delta_2)^{s_D} \quad (18c)$$

$$(x_j^i)'' \in [0, (u_j^i)''], \quad \forall (i,j) \in I_d, 1 \leq d \leq (2/\delta_2)^{s_D} \quad (18d)$$

Observe that (LP₇) and (LP₈) satisfy the structure of the LP obtained in Step 2 of Theorem 3 and Theorem 1 respectively, so we can directly apply the rounding procedures there.

Step 3: Round these $\mathcal{O}(s_D \tau)$ fractional variables.

The rounding procedure in this step also follows straightforwardly from applying Step 3 of Theorem 3 and Theorem 1 to (LP₇) and (LP₈), since the two parts $z_{i\phi}$ and $(x_j^i)''$ are relatively independent. Hence, it brings about a total error of $\mathcal{O}(s_D \tau) \cdot \max\{\delta_1 \|\mathcal{D}^i\|_{\infty}, \delta_2 \Delta_2\}$ to Constraint (16b). Note that $\mathcal{O}(s_D \tau) \cdot \max\{\delta_1 \|\mathcal{D}^i\|_{\infty}, \delta_2 \Delta_2\} \leq \mathcal{O}(s_D \tau) \cdot \max\{\delta_1, \delta_2\} \cdot b_j^0$ in the j -th dimension. We can take $\delta_1 = \frac{1}{s_D} (\frac{\varepsilon}{s_A t})^{\mathcal{O}(t)}$, $\delta_2 = \mathcal{O}(\frac{\varepsilon}{s_D})$ to make $\mathcal{O}(s_D \tau) \cdot \max\{\delta_1, \delta_2\} \leq \varepsilon$. Therefore, we have $(1 - \varepsilon) \mathbf{b}^0 \leq \sum_{i=1}^n D^i \mathbf{x}^i \leq (1 + \varepsilon) \mathbf{b}^0$.

Running time.

$$\text{Step 1: } 2^{\left(2^{(s_D \cdot (s_A t / \varepsilon)^t)^{\mathcal{O}(1)}} + (s_D / \varepsilon)^{\mathcal{O}(s_D)}\right)^{\mathcal{O}(1)}} \cdot |I|$$

$$\text{Step 2: } (\rho n)^4 \cdot |I| = 2^{(s_D \cdot (s_A t / \varepsilon)^t)^{\mathcal{O}(1)}} n^4 \cdot |I|$$

$$\text{Step 3: } \rho \tau^2 n^2 \log n + (2/\delta_2)^{s_D} \cdot n \log n = \left(2^{(s_D \cdot (s_A t / \varepsilon)^t)^{\mathcal{O}(1)}} + (s_D / \varepsilon)^{\mathcal{O}(s_D)}\right)^{\mathcal{O}(1)} \cdot n^2 \log n$$

All in all, the total time is $2^{\left(2^{(s_D \cdot (s_A t / \varepsilon)^t)^{\mathcal{O}(1)}} + (s_D / \varepsilon)^{\mathcal{O}(s_D)}\right)^{\mathcal{O}(1)}} \cdot \text{poly}(|I|)$, which has the form of $f(s_A, s_D, t, \varepsilon) \text{poly}(|I|)$. \blacktriangleleft