



The Support of Bin Packing Is Exponential

Klaus Jansen  

Department of Computer Science, Kiel University, Germany

Lis Piroton  

Department of Computer Science, Kiel University, Germany

Malte Tutas  

Department of Computer Science, Kiel University, Germany

Abstract

Consider the classical Bin Packing problem with d different item sizes s_i and amounts of items a_i . The support of a Bin Packing solution is the number of differently filled bins. In this work, we show that the lower bound on the support of this problem is $2^{\Omega(d)}$. Our lower bound matches the upper bound of 2^d given by Eisenbrand and Shmonin [Oper. Research Letters '06] up to a constant factor. This result has direct implications for the time complexity of several Bin Packing algorithms, such as Goemans and Rothvoss [SODA '14], Jansen and Klein [SODA '17] and Jansen and Solis-Oba [IPCO '10].

To achieve our main result, we develop a technique to aggregate equality constrained ILPs with many constraints into an equivalent ILP with one constraint. Our technique contrasts existing aggregation techniques as we manage to integrate upper bounds on variables into the resulting constraint. We believe this technique can be useful for solving general ILPs or the d -dimensional knapsack problem.

2012 ACM Subject Classification Theory of computation → Integer programming; Theory of computation → Packing and covering problems

Keywords and phrases Bin Packing, Integer Programming, Support

Digital Object Identifier 10.4230/LIPIcs.ESA.2025.48

Funding This work was supported by the German Research Foundation (DFG) – Project DFG JA 612/27-1.

1 Introduction

In their seminal work [8] Eisenbrand and Shmonin inspect integer conic combinations $b \in \mathbb{Z}^d$ of a finite set of integer vectors $X \subset \mathbb{Z}^d$. They provide upper bounds on the size of the smallest subset $X^* \subseteq X$ such that b is an integer conic combination of elements in X^* . This can be interpreted in the context of integer programming. Integer Programs (IPs) are composed of a matrix $A \in \mathbb{Z}^{d \times n}$, an (optional) cost vector $c \in \mathbb{Z}^n$, a target vector $b \in \mathbb{Z}^d$ and a solution vector $x \in \mathbb{Z}^n$. The goal is to compute the solution vector satisfying $Ax = b$ while minimizing the value $c^T x$. In this context, the set of integer vectors correspond to the columns of A and the size of the smaller subset X^* is the number of non-zero entries (the *support*) in the solution vector. More precisely, they show that for any polytope $P \subseteq \mathbb{R}^d$ and any integral vector $x \in \mathbb{Z}_{\geq 0}^{|P \cap \mathbb{Z}^d|}$ of multiplicities there exists a $x^* \in \mathbb{Z}_{\geq 0}^{|P \cap \mathbb{Z}^d|}$ such that $|\text{supp}(x^*)| \leq 2^d$ and $\sum_{p \in P \cap \mathbb{Z}^d} x_p \cdot p = \sum_{p \in P \cap \mathbb{Z}^d} x_p^* \cdot p$. They achieve this through a sophisticated exchange argument, transforming a solution with a large support into one with a bounded support. They show a second bound of $O(d \log(d\Delta))$, where $\Delta = \|A\|_\infty$ is the largest number in the matrix A , in a similar fashion. These celebrated result have found application in a variety of applications. These include classical problems of computer science like scheduling on identical and uniform machines (using the second bound) [15, 19] and bin packing problem (using the first bound) [13, 18, 20].



© Klaus Jansen, Lis Piroton, and Malte Tutas;
licensed under Creative Commons License CC-BY 4.0

33rd Annual European Symposium on Algorithms (ESA 2025).

Editors: Anne Benoit, Haim Kaplan, Sebastian Wild, and Grzegorz Herman; Article No. 48; pp. 48:1–48:16

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

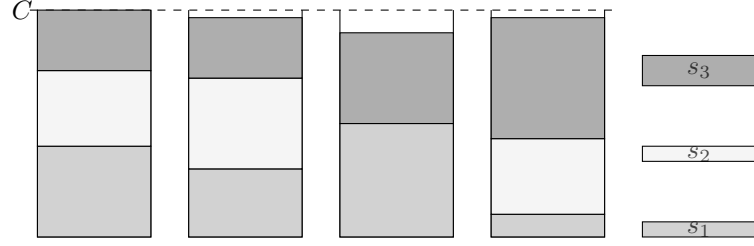


Figure 1 An illustration of the BIN PACKING problem with three item types, four bins and capacity C . Colors indicate item types. Each bin has a unique *configuration*, i.e., combination of items, assigned to it. Only the leftmost bin is completely filled. The sizes of each item type are shown on the right.

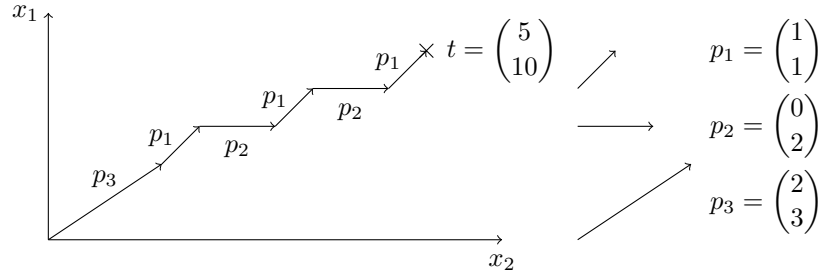


Figure 2 An illustration of the CONE AND POLYTOPE INTERSECTION problem with three points $p_i \in P$ and the target $t \in Q$. The values of each vector are given. A possible solution is shown.

Improvements on the first bound would have direct implications for the time complexity of several algorithms. Consider the high-multiplicity BIN PACKING (BP) problem with bin size C : We are given item sizes $s_1, \dots, s_d \in [0, C]$, and multiplicities $a_i \in \mathbb{Z}_{\geq 0}, i \in [d]$. The task is to find the minimum number $B \in \mathbb{N}$ such that all items can be packed into B bins of size C . See Figure 1 for an illustration. For this problem, Jansen and Solis-Oba [20] give an additive approximation algorithm with guarantee $\text{OPT}+1$. The time complexity of their algorithm is $d^{O(d^2)} \cdot 2^{O(8^d)} \cdot \text{poly}(\text{enc})$, where $\text{poly}(\text{enc})$ represents a polynomial in the input. In their paper, they ask whether the support bound by Eisenbrand and Shmonin can be improved to be polynomial in d . Consequently, this would reduce the dependency on d in the algorithm to be singly-exponential, i.e., result in a running time of $2^{\text{poly}(d)} \cdot \text{poly}(\text{enc})$.

Following this result, Goemans and Rothvoss [13] provide an algorithm running in time $\text{enc}(s, a, C)^{2^{O(d)}}$. Here, $\text{enc}(s, a, C)$ denotes the bitsize of encoding the size vector s , the multiplicity vector a and the bin capacity C in binary. The time complexity of this algorithm would also be improved by a smaller support bound. For example, a polynomial support $\text{poly}(d)$ in d directly improves their running time to $\text{enc}(s, a, C)^{\text{poly}(d)}$. They achieve this result by solving a more general problem, the CONE AND POLYTOPE INTERSECTION (CAPI) problem. Here, we are given two polytopes $P, Q \subseteq \mathbb{R}^d$, where P is bounded. The task is to decide whether there is a point in Q that can be expressed as a non-negative integer combination of integer points in P . Goemans and Rothvoss show that this problem captures the essence of high-multiplicity BP. They reduce BP to CAPI by setting the integer points in $P = \left\{ \begin{pmatrix} x \\ 1 \end{pmatrix} \in \mathbb{R}_{\geq 0}^{d+1} \mid s^T x \leq C \right\}$ to every possible configuration, i.e., combinations of items that fit in a single bin. They set $Q = \begin{pmatrix} a \\ B \end{pmatrix}$ to be the point that corresponds to all items being taken. The final entry in P and Q is used to count the number of used bins B .

Jansen and Klein extend the work of Goemans and Rothvoss in [18]. They give an algorithm for bin packing that depends on the number V_I of vertices in the underlying knapsack polytope of a single bin. Their algorithm has a running time of $|V_I|^{2^{O(d)}} \text{enc}(s, a, C)^{O(1)}$. Thus, it improves upon Goemans and Rothvoss' result when V_I is small. However, it does not improve upon the result by Goemans and Rothvoss in the general sense, as the bound on V_I is $\Omega(\text{enc}(s, a, C))^d$ as given by Hayes and Larman [14].

All three of these results could be improved with a polynomial (in d) support bound for BP. However, our result shows that there is no polynomial support bound in d for bin packing, and thus answers the question posed by Jansen and Solis-Oba negatively. Goemans and Rothvoss made first strides in this direction in [13], where they show that the bound given by Eisenbrand and Shmonin is tight for polytopes, i.e., there is an exponential support when reaching a target with only column vectors or linear combinations of them. They construct a set of column vectors X with dimension d and $|X| = 2^{d-1}$. Define the polytope $P = \text{conv}(X)$, where $\text{conv}(X)$ is the convex hull of X . They then define $t = \sum_{x \in X} x$ and show that you can only reach t by taking every element of X exactly once. This shows a lower bound on the support of $2^{d-1} = 2^{\Omega(d)}$. However, their bound does not admit a knapsack constraint and can therefore not be applied to the bin packing problem directly. This is because the columns in the bin packing ILP originate from a knapsack constraint that determines the feasible packings of a single bin.

With so many different problems gaining important information from support bounds, it should come as no surprise that there are other bounds on the support of integer programs with different parameterizations. Berndt, Brinkop, Jansen, Mnich and Stamm show that the support is also bounded by $d \cdot (\log(3\|A\|_1) + \sqrt{\log(\|A\|_1)})$ [3]. They complement this result by showing a lower bound on the size of the support of $d \cdot (\lfloor \log(\|A\|_1) \rfloor + 1)$. Aliev, de Loera, Oertel and O'Neill [2] give two support bounds for linear diophantine equations of $r(A) + \lfloor \log(H(A)) \rfloor$, where $r(A)$ is the column rank of the matrix A and $H(A)$ is the determinant of the span of the $r(A)$ -dimensional sublattice of \mathbb{Z}^n formed by all integer points in the linear subspace spanned by the columns of A , i.e., $H(A) = \det(\text{span}_{\mathbb{R}}(A) \cap \mathbb{Z}^n)$. Their second bound is $\lfloor 2d \log(2\sqrt{d}\Delta) \rfloor$. Aliev, De Loera, Eisenbrand, Oertel and Weismantel [1] show the bound of $\lfloor 2d \log(2\sqrt{d}\Delta) \rfloor$ for general ILPs.

The main tool we use in the following is *aggregation*. Aggregation is a powerful tool that transforms an ILP with multiple constraints into a single constraint. Research into this technique began in the 70's, resulting in several different aggregation methods, see e.g. [4, 5, 9, 11, 12, 21, 22, 23, 24]. Elmaghraby and Wig [9] iteratively combine two equations into a single one. Applying this technique repeatedly yields a single constraint. Another approach is given by Padberg [22]. Instead of combining two equations iteratively, he aggregates all equations at the same time.

Our approach is based on the techniques by Padberg [22]. Our aggregation techniques also aggregate all constraints in a single step. However, we manage to include the upper bounds of the variables inside the resulting knapsack constraints, where Padbergs method leaves them outside the constraint. We believe that this type of aggregation can find application in different contexts, such as d -dimensional knapsack problems or general ILPs with d constraints. Here, the aggregation may yield faster algorithms, especially when one manages to reduce the size of the coefficients. This may be done by using techniques by Frank and Tardos [10], Eisenbrand, Hunkenschröder, Klein, Koutecký Levin and Onn [7] or Jansen, Kahler and Zwanger [16].

On the negative side, it is known that aggregation is not feasible for a set of inequalities, as was proven by Chvátal and Hammer [6]. They provide a simple two-line ILP that cannot be aggregated.

1.1 Our Contribution

In this work, we continue research into the support of BIN PACKING solutions. We show that a set of points of CONE AND POLYTOPE INTERSECTION can be embedded inside a knapsack polytope of dimension $O(d)$. Using this polytope, we show that the support of the high-multiplicity BP problem is exponential, requiring at least $2^{\Omega(d)}$ many different configurations in an optimum solution. This yields our main result, Theorem 1:

► **Theorem 1.** *There exists a high-multiplicity BIN PACKING instance of dimension d such that the solution-vector x (a vector of configurations) contains at least $2^{\Omega(d)}$ non-zero entries, i.e., $|\text{supp}(x)| \in 2^{\Omega(d)}$.*

With this result, we answer an open question posed by Jansen and Solis-Oba in [20]. They provide an algorithm for the cutting stock problem. The running time of this algorithm could be improved to be singly-exponential if the support of bin packing is not exponential. However, our result answers this question negatively. In a similar vein, our result shows that the algorithm by Goemans and Rothvoss [13] can not be directly improved. The same holds for the algorithm by Jansen and Klein [17].

A major tool to achieve this result is the aggregation of multiple constraints without upper bounds on the variables into a single constraint. To the best of our knowledge, this type of constraint-aggregation has not been done before. Contrasting prior results, our method of aggregation manages to include upper bounds inside the constraints instead of copying them from the original problem. This technique is given in Section 3.

Overview

Here, we give a brief overview of the structure of the document and the structure of our main proof. We begin, in Section 3, by developing a technique with which we can aggregate an ILP containing only d equalities into one containing only a single equality. Next, in Section 4.1 we adapt an idea originally given by Goemans and Rothvoss in [13] to construct a polytope P of dimension $d' + 1$ and generate a target vector t . We provide an alternative proof to [13] that shows we require $2^d - 1$ points in P to reach t . Next, in Section 4.2, we construct an equality ILP with $O(d)$ constraints. Each constraint later defines an item size of the BP problem. We show that there are $2^d - 1$ feasible solutions to this ILP, with each solution x having the first $d + 1$ entries as vectors in P , i.e., there is one solution to the ILP for each vector in P . These solutions are the $2^d - 1$ configurations X^* of the BP problem. Then, we aggregate this ILP using the technique in Section 3. This yields an equality ILP with one row. We transform this into a knapsack constraint with right hand side C . Denote the set of all configurations with total size at most C as X . This constraint is then used to define the capacity of a bin in the BP problem. The total number of items of each type is given as the sum of all configurations in X^* . To place all these items in the optimal $2^d - 1$ bins, we know, by Section 4.1, that need to take every configuration in X^* exactly once. Taking any configuration in $X \setminus X^*$ would require at least one additional bin. This proves Theorem 1.

2 Preliminaries

For a positive integer n , we define $[n] := \{1, 2, \dots, n\}$ and $[n]_0 := \{0, 1, \dots, n\}$. A vector that contains a certain value in all entries is stylized, e.g. the zero vector 0_d is written as \mathbb{O}_d . We omit the dimension d if it is apparent from the context. The support of a d -dimensional vector v is the set of its indices with a non-zero entry. It is denoted by $\text{supp}(v) := \{i \in [d] \mid v_i \neq 0\}$. We define the size of the support with $s(v) := |\text{supp}(v)|$.

► **Definition 2** (Bin Packing). *Given is a set of d different item types with sizes $s_i \in (0, C]$ and amounts $a_i \in \mathbb{Z}_{\geq 0}$. Given a number $B \in \mathbb{Z}_{\geq 0}$, the goal is to decide whether all items can be assigned to one of B bins, while all bins are filled to at most C .*

A powerful tool to model countless optimization problems is that of integer programming.

► **Definition 3** (Integer Programming). *An integer linear program (ILP) is composed of a matrix $A \in \mathbb{Z}^{d \times n}$, an (optional) cost vector $c \in \mathbb{Z}^n$, a target vector $b \in \mathbb{Z}^d$ and a solution vector $x \in \mathbb{Z}_{\geq 0}^n$. The goal is to compute a solution vector satisfying $Ax = b$ while minimizing the value $c^T x$.*

Such integer programs can be used to express high-multiplicity BIN PACKING. These are often referred to as *configuration ILPs*. A configuration is a feasible assignment $x \in \mathbb{Z}_{\geq 0}^d$ of items into a bin, i.e., one with total size at most $s^T x \leq C$. Denote the set of all feasible configurations as \mathcal{C} , and set $n := |\mathcal{C}|$. Set $c = \mathbb{1}_n$ and let each column of A represent a configuration, where row i corresponds to item type i . Finally set the right hand side $b := a$, i.e., the vector representing the total number of items of each size in the instance.

A useful tool that we use is the classical knapsack problem.

► **Definition 4** (Unbounded Knapsack). *Given a set I of n items each defined by a profit $p_i \in \mathbb{R}_{\geq 0}$ and a weight $w_i \in \mathbb{Z}_{\geq 1}$, along with a knapsack of capacity $C \in \mathbb{Z}_{\geq 1}$. Find a solution vector $x \in \mathbb{Z}_{\geq 0}^n$ such that $\sum_{i=0}^n w_i x_i \leq C$ and $\sum_{i=0}^n p_i x_i$ is maximal. We call $\sum_{i=0}^n w_i x_i = C$ the equality Knapsack constraint to the corresponding problem.*

Following this definition, we call the set of all solutions fulfilling $x_i \geq 0, \forall i \in [n]$ and $\sum_{i=0}^n w_i x_i \leq C$ the Knapsack Polytope.

3 Aggregation of an ILP

In this section, we show how to transform an ILP with multiple equality constraints into a single knapsack constraint. The novel aspect of our technique is that we also include the external upper bounds of the variables in the aggregation.

Consider an ILP of the form $\min\{c^T x \mid Ax = b, x \in \mathbb{Z}_{\geq 0}^n, x \leq u\}$ with $A = (a_{ij})_{i \in [d], j \in [n]} \in \mathbb{Z}^{d \times n}$, $c \in \mathbb{Z}^n$, $u \in \mathbb{Z}_{\geq 0}^n$ and $b \in \mathbb{Z}^d$ and let $\Delta := \|A\|_{\infty}$ be the largest absolute value in A .

For each variable $x_j, j \in [n]$, we define the following constraint: $x_j + s_j = u_j$, where $s_j \in \mathbb{Z}_{\geq 0}$ is a slack variable. This simulates the upper bound $x_j \leq u_j$. Define $U := \sum_{j=1}^n u_j$ as the overall upper bound on the sum of all variables. For this, add the constraint:

$$\sum_{i=j}^n (x_j + s_j) + s_{n+1} = U, \quad (1)$$

with $s_{n+1} \in \mathbb{Z}_{\geq 0}$ as a slack variable. Note that bounding the sum of variables by the sum of their upper bounds does not change the solution.

This results in a new ILP $A' \begin{pmatrix} x \\ s \end{pmatrix} = (b, u, U)$, where

$$A' := \begin{pmatrix} A & 0_{dn} & 0 \\ I_n & I_n & 0 \\ \mathbb{1}_n & \mathbb{1}_n & 1 \end{pmatrix} \in \mathbb{Z}^{(d+n+1) \times (2n+1)}.$$

48:6 The Support of Bin Packing Is Exponential

Next, we use the bound U to define a large integer M , which we then use to aggregate the constraints.

$$\begin{aligned} M &:= \Delta U + \max(\|b\|_\infty, \|u\|_\infty) + \Delta + 2 \\ &> \Delta U + \max(\|b\|_\infty, \|u\|_\infty). \end{aligned} \quad (2)$$

By giving the upper-bound constraint (1) the largest weight M^{d+n} , we ensure that it may never be broken and thus, keeps the sum of the variables in its range. Multiplying both sides of the equations with $1, M, M^2, \dots, M^{d+n}$ generates the following ILP where all variables have to be non-negative integers.

$$\begin{aligned} \sum_{j=1}^n a_{1j}x_j &= b_1 \\ M \sum_{j=1}^n a_{2j}x_j &= Mb_2 \\ &\vdots \\ M^{d-1} \sum_{j=1}^n a_{mj}x_j &= M^{d-1}b_m \\ M^d(x_1 + s_1) &= M^d u_1 \\ &\vdots \\ M^{d+n-1}(x_n + s_n) &= M^{d+n-1}u_n \\ M^{d+n}(\sum_{j=1}^n (x_j + s_j) + s_{n+1}) &= M^{d+n}U \\ x_j &\in \mathbb{Z}_{\geq 0} \quad \forall j \in [n] \\ s_j &\in \mathbb{Z}_{\geq 0} \quad \forall j \in [n+1] \end{aligned} \quad (3)$$

These constraints can also be formulated as $A' \begin{pmatrix} x \\ s \end{pmatrix} = (b, u, U)$, where

$$A' := \begin{pmatrix} A & 0_{dn} & 0 \\ I_n & I_n & 0 \\ \mathbf{1}_n & \mathbf{1}_n & 1 \end{pmatrix} \in \mathbb{Z}^{(d+n+1) \times (2n+1)}$$

An intuitive view of the aggregation technique is that the resulting single constraints solution is a base M number, where M is a large integer. There, the smallest digit represents the solution of the first constraint, the second constraints solution is represented by the second smallest digit and so on.

Next, we show that this ILP can be aggregated into a single equality knapsack constraint by proving that their solutions are identical. The proof is presented in the full version.

► **Lemma 5** (\asymp). *The vector $\text{sol} = (x, s)^T \in \mathbb{Z}_{\geq 0}^{2n+1}$ is a feasible integer solution to (3) if and only if sol is an integer solution to the following equation:*

$$\begin{aligned} \sum_{i=1}^d (M^{i-1} \sum_{j=1}^n (a_{ij}x_j)) + \sum_{j=1}^n (M^{d+j-1}(x_j + s_j)) + M^{d+n}(\sum_{j=1}^n (x_j + s_j) + s_{n+1}) \\ = \sum_{i=1}^d (M^{i-1}b_i) + \sum_{j=1}^n (M^{d+j-1}u_j) + M^{d+n}U \end{aligned} \quad (4)$$

4 Bounding the Support for Bin Packing

Assume we are given a knapsack polytope P of dimension $d' + 1$. Also assume that there exists a BIN PACKING instance \mathcal{I} with just one unique integer solution $x \in \mathbb{Z}_{\geq 0}^d$, $d = O(d')$ to the CONFIGIP and the $d' + 1$ -dimensional integer points in P are equal to the first $d' + 1$

dimensions of the configurations. If we now require k different points in P to represent a vector t as a conic integer combination then x has at least k non-zero entries, which means that we need at least k different configurations in the BIN PACKING solution. If now k is exponential in d' , we have shown that there exists an exponential lower bound on the support of the BIN PACKING problem, i.e., Theorem 1.

4.1 The support of closed polytopes

In this section, we give a knapsack polytope P of dimension $d+1$ and show that there exists a vector $t \in \mathbb{Z}^{d+1}$ such that we need at least $2^d - 1$ points in P to represent t as a conic integer combination. We adapt a construction pioneered by Goemans and Rothvoss in [13] and set

$$\begin{aligned} X &= \{\mathbf{0}\} \cup \{(x_1, \dots, x_d, (4k)^{\sum_{i=0}^{d-1} 2^i x_i}) \mid x_i \in \{0, 1\} \forall i \in [d]\} \setminus \{(0, \dots, 0, 1)\} \\ &= \{a_0, a_1, \dots, a_k\} \end{aligned} \quad (5)$$

with $k = 2^d - 1$ and define $P := \text{conv}(X)$. E.g. for $d = 2$, we have

$$X = \left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 4k \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ (4k)^2 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ (4k)^3 \end{pmatrix} \right\}.$$

Assume that the elements in X are sorted according to their length, i.e., $\|a_0\|_\infty = 0$ and $\|a_i\|_\infty = (4k)^i$ for all $i \in [k]$. Then the following lemma proves the goal of this section.

► **Lemma 6.** *Set $t = \sum_{i=0}^k a_i$. The conic combination $t = \sum_{i=1}^k a_i$ is unique (except for arbitrarily adding a_0). This implies that we need all $2^d - 1$ points in P exactly once to represent t as a conic integer combination.*

Proof. Each integer conic combination that represents t and includes $\mathbf{0}$ can be transformed into an integer conic combination of less points by removing $\mathbf{0}$. Therefore, we can assume that $\mathbf{0}$ is not part of the conic combination.

Now considering P , the only integer points in P are the elements in X , i.e., $P \cap \mathbb{Z}^{d+1} = X$. This holds due to the first d coordinates. For all $\mu_i \in (0, 1)$ with $\sum_i \mu_i = 1$ it holds that $\mu_i \cdot 0 + \mu_j \cdot 1 \notin \mathbb{Z}$ and for each pair of vectors $(a_i, a_j) \in X$ with $i \neq j$, there is at least one coordinate ℓ , where with $a_i(\ell) = 0$ and $a_j(\ell) = 1$ or $a_i(\ell) = 1$ and $a_j(\ell) = 0$. Thus, there can not be any integer points other than X in $P \cap \mathbb{Z}^{d+1}$.

$$\text{Now consider an arbitrary integer conic combination } \sum_{i=1}^k \mu_i a_i = t = \begin{pmatrix} 2^{d-1} \\ \vdots \\ 2^{d-1} \\ \sum_{i=1}^k (4k)^i \end{pmatrix}.$$

Since $t_1 = \dots = t_{d-1} = 2^{d-1}$ and we do not include $\mathbf{0}$, we can not take a single vector more than $2^{d-1} \leq k$ times. Furthermore, we can not take more than dk vectors total, as each vector except $\mathbf{0}$ has an entry in at least one of the d dimensions, and each entry can only sum up to k . This leaves us with the following:

$$\begin{aligned} 0 \leq \sum_{i=1}^k \mu_i &\leq d \cdot 2^{d-1} \leq dk, \quad \text{and} \\ \mu_i &\leq 2^{d-1} \leq k \quad \forall i \in [k] \end{aligned}$$

i.e., that at most dk vectors can be summed up to reach t and each vector is taken $\leq k$ times.

We now show that $\mu_i = 1$ for all $i \in [k]$ is the only solution to $t = \sum_{i=1}^k \mu_i a_i$. First, observe that $t_{d+1} = \sum_{i=1}^k (4k)^i < 2(4k)^k$ holds. Therefore, we have $\mu_k < 2$. Next, observe that $k(4k)^{x-1} < (4k)^x - (4k)^{x-1} \Leftrightarrow (k+1)(4k)^{x-1} < (4k)^x$ holds for all $x \in \mathbb{Z}_{\geq 1}$ because $(k+1) < 4k$. Because of this, even after adding the maximum possible k copies of the $k-1$ -th item, we have that $t_{d+1} - k(4k)^{k-1} > (4k)^{k-1}$. We can only compensate this difference by adding smaller items. Here, we can apply the argument iteratively, as we can only add k copies of column $(k-2)$ to sum up to a value $> (4k)^{k-1}$. Thus, $\sum_{i=1}^{k-1} k(4k)^i < (4k)^k$ holds. Therefore, we can not sum up to $(4k)^k$ without the last item and $\mu_k > 0$ directly follows. Taking $\mu_k < 2$ and $\mu_k > 0$ together, we have $\mu_k = 1$. Because of $\mu_k = 1$, we have $t_{d+1} = \sum_{i=1}^k -1(4k)^i$, and we only need to consider $\mu_i, i \in [k-1]$. Then, this argument can be applied iteratively, yielding $\mu_i = 1$ for all $i \in [k]$. \blacktriangleleft

By Lemma 6, we have shown that we require $2^d - 1$ different points in P to represent a vector t as a conic integer combination. In the following we build the bridge to BIN PACKING and prove that there exists a BIN PACKING instance where the first $d+1$ dimensions of the solution vector of its CONFIGIP equals the integer points in P . In total, our BIN PACKING instance has $O(d)$ dimensions which then with Lemma 6 implies the exponential lower bound of BIN PACKING and therefore Theorem 1.

4.2 Construction of the ILP

In this section, we now aim to construct an ILP with equality constraints, where the first $d+1$ coordinates in its solution vectors are unique and form the following set:

$$\{(x_0^{\text{bin}}, \dots, x_{d-1}^{\text{bin}}, \gamma^{\sum_{\ell=0}^{d-1} 2^\ell x_\ell^{\text{bin}}}) \mid x_\ell^{\text{bin}} \in \{0, 1\} \forall \ell \in [d-1]_0 \setminus \{(0, \dots, 0, 1)\}\}. \quad (6)$$

Note that this is exactly the set X of points in P in the prior section without $a_0 = \mathbf{0}$ and where $\gamma = 4k$ and that the solution vectors form the configurations in the CONFIGIP.

Also note that the last coordinate is of the form γ^i for some $i \in \mathbb{Z}_{\geq 1}$ and the first d coordinates are the binary encoding of i . We first define a variable $r(d)$, which we use to construct a set of constraints that is only feasible if $r(d) = \gamma^i$ holds for some $i \in \mathbb{Z}_{\geq 1}$. Additionally, a certain set of variables in the constraints matches the points in Equation (6). In second part of this section, we use the aggregation presented in Section 3 to transform these constraints into a single knapsack constraint. We provide a proof that the knapsack constraint is indeed equivalent to the proposed set of constraints (i.e., both allow the same set of solutions) and requires $O(d)$ many variables. With this equivalence, we then have that the first $d+1$ dimensions of the configurations in the CONFIGIP form the set Equation (6). In combination with the Lemma 6, this then implies that there exists an exponential lower bound on the support of the BIN PACKING problem.

Next, assume that we are given a dimension $d \in \mathbb{Z}_{\geq 1}$ and a base $\gamma \in \mathbb{Z}_{\geq 2}$. We introduce a set of constraints with $O(d)$ variables which has exactly $2^d - 1$ integer solutions. For each $i \in [2^d - 1]$, the variable $r(d)$ of the solution vector is forced to take the value γ^i . Additionally, the variables $x^{\text{bin}}(\ell) \in \{0, 1\}, \ell \in [d-1]_0$ match the binary encoding of i , i.e., $i = \sum_{\ell=0}^{d-1} 2^\ell x^{\text{bin}}(\ell)$.

For a brief overview, our constraints behave as follows. First, the value $r(d)$ is some integer in the interval $[2, \gamma^{2^d-1}]$. Let us assume that $r(d) = \gamma^i$ for some i . Later, we show that we only allow those integers that equal γ^i where i is a positive integer. Next, for each $\ell = d-1, \dots, 0$, we introduce the variable $y(\ell)$ that helps us to correctly determine the bit $x^{\text{bin}}(\ell)$ of the binary encoding. We also introduce the variable $z(\ell)$ that helps us to determine

the $r(\ell)$ which will then be used recursively. For a more detailed version of the procedure of converting a decimal number $i \in \mathbb{Z}_{\geq 1}$ to binary, we refer to Algorithm 1. Note that the number we aim to convert is in the exponent of γ .

■ **Algorithm 1** Decimal to Binary.

Input: Dimension $d \in \mathbb{Z}_{\geq 1}$; Base $\gamma \in \mathbb{Z}_{\geq 2}$; Exponent $i \in \mathbb{Z}_{\geq 1}$

Output: Binary encoding x^{bin} of i

- 1: Set $r(d) := \gamma^i$ with $i \in \mathbb{Z}_{\geq 1}$ and $\gamma^i \leq \gamma^{2^d-1}$.
 - 2: **for** $\ell = d - 1, \dots, 0$ **do**
 - 3: **if** $r(\ell + 1) \geq \gamma^{2^\ell}$ **then**
 - 4: Record a 1 in position ℓ , i.e., $x^{\text{bin}}(\ell) := 1$.
 - 5: Subtract 2^ℓ from the current exponent, i.e., $r(\ell) := \frac{r(\ell+1)}{\gamma^{2^\ell}}$.
 - 6: **else**
 - 7: Record a 0 in position ℓ , i.e., $x^{\text{bin}}(\ell) := 0$.
 - 8: Do not change the current exponent, i.e., $r(\ell) := r(\ell + 1)$.
 - return** the vector $x^{\text{bin}}(\ell), \ell \in [d - 1]_0$
-

The following set of constraints simulates this algorithm. Let α be some integer number larger than 3. Later, α represents the total number of constraints. Each set of constraints $\mathcal{C}_{\ell,k}, k \in [13]$ is defined for all $\ell \in [d - 1]_0$. They simulate the algorithm above. Additionally, $\mathcal{C}_{\alpha-2}$ bounds $r(d)$, i.e., ensures its binary encoding has at most d bits. \mathcal{C}_0 guarantees that in the end, there is no remainder left. This constraint is also needed to guarantee that the exponent i is integer (Lemma 12). Also, we do not want to allow $i = 0$, respectively $x^{\text{bin}}(\ell) \neq 0$. That is done by $\mathcal{C}_{\alpha-3}$. Thus, in total, we have $13d + 3$ constraints.

► **Constraints 1.** For each $\ell \in [d - 1]_0$, we have the following set of constraints:

$$\begin{array}{ll}
 y(\ell) & \geq 0 & (\mathcal{C}_{\ell,1}) \\
 y(\ell) & \leq (r(\ell+1)/\gamma^{2^\ell}) + 1/(\gamma^{2^\ell}+1) & (\mathcal{C}_{\ell,2}) \\
 y(\ell) & \geq (r(\ell+1)/\gamma^{2^\ell}) - (\gamma^{2^\ell}-1)/\gamma^{2^\ell} & (\mathcal{C}_{\ell,3}) \\
 x^{\text{bin}}(\ell) & \geq 0 & (\mathcal{C}_{\ell,4}) \\
 x^{\text{bin}}(\ell) & \leq 1 & (\mathcal{C}_{\ell,5}) \\
 x^{\text{bin}}(\ell) & \leq y(\ell) & (\mathcal{C}_{\ell,6}) \\
 y(\ell) & \leq (\gamma^{2^\ell} + 1)x^{\text{bin}}(\ell) & (\mathcal{C}_{\ell,7}) \\
 r(\ell) & \geq 0 & (\mathcal{C}_{\ell,8}) \\
 (\gamma^{2^\ell} - 1)z(\ell) + r(\ell) & = r(\ell + 1) & (\mathcal{C}_{\ell,9}) \\
 z(\ell) & \geq 0 & (\mathcal{C}_{\ell,10}) \\
 z(\ell) & \geq -\gamma^{2^\ell} + \gamma^{2^\ell} x^{\text{bin}}(\ell) + r(\ell) & (\mathcal{C}_{\ell,11}) \\
 z(\ell) & \leq \gamma^{2^\ell} x^{\text{bin}}(\ell) & (\mathcal{C}_{\ell,12}) \\
 z(\ell) & \leq r(\ell) & (\mathcal{C}_{\ell,13})
 \end{array}$$

Also, we add the additional constraints $\mathcal{C}_0, \mathcal{C}_{\alpha-3}$ and $\mathcal{C}_{\alpha-2}$

$$\begin{array}{ll}
 r(0) & = 1 & (\mathcal{C}_0) \\
 r(d) & \geq 2 & (\mathcal{C}_{\alpha-3}) \\
 r(d) & \leq \gamma^{2^d-1} & (\mathcal{C}_{\alpha-2})
 \end{array}$$

All of the constraints are necessary to ensure that $r(d)$ takes some value of the form $\gamma^i, i \in \mathbb{Z}_{\geq 1}$, while x^{bin} is the binary encoding of i . Moreover, the constraints forbid any other assignment of $r(d)$ and x^{bin} . All other variables are forced to take a specific value based on i . In the full version, we give an example that we cannot omit the variables $z(\ell)$, as that would allow other solutions with $r(d) = \gamma^i, i \in \mathbb{Z}_{\geq 1}$.

48:10 The Support of Bin Packing Is Exponential

For easier understanding, we now present an example.

► **Example 7.** Let $\gamma \in \mathbb{Z}_{\geq 2}$ and set $d := 3$. Then, for $i \in [7]$, we get the following values for the variables.

i	1	2	3	4	5	6	7
$r(3)$	γ^1	γ^2	γ^3	γ^4	γ^5	γ^6	γ^7
$y(2)$	0	0	0	1	γ^1	γ^2	γ^3
$y(1)$	0	1	γ^1	0	0	1	γ^1
$y(0)$	1	0	1	0	1	0	1
$x^{\text{bin}}(2)$	0	0	0	1	1	1	1
$x^{\text{bin}}(1)$	0	1	1	0	0	1	1
$x^{\text{bin}}(0)$	1	0	1	0	1	0	1
$r(2)$	γ^1	γ^2	γ^3	$\gamma^{4-4} = 1$	$\gamma^{5-4} = \gamma^1$	$\gamma^{6-4} = \gamma^2$	$\gamma^{7-4} = \gamma^3$
$r(1)$	γ^1	$\gamma^{2-2} = 1$	$\gamma^{3-2} = \gamma^1$	1	γ^1	$\gamma^{2-2} = 1$	$\gamma^{3-2} = \gamma^1$
$r(0)$	$\gamma^{1-1} = 1$	1	$\gamma^{1-1} = 1$	1	$\gamma^{1-1} = 1$	1	$\gamma^{1-1} = 1$
$z(2)$	0	0	0	1	γ^1	γ^2	γ^3
$z(1)$	0	1	γ^1	0	0	1	γ^1
$z(0)$	1	0	1	0	1	0	1

Now, we state properties for the variables $y(\ell)$, $x^{\text{bin}}(\ell)$, $r(\ell)$, $z(\ell)$ based on the value of $r(\ell + 1)$. After that, we prove that the vector x^{bin} is indeed the binary encoding of i .

$\mathcal{C}_{\ell,1}$, $\mathcal{C}_{\ell,4}$, $\mathcal{C}_{\ell,8}$ and $\mathcal{C}_{\ell,10}$ are lower bounds of the variables $y(\ell)$, $x^{\text{bin}}(\ell)$, $r(\ell)$ and $z(\ell)$. We omit these constraints later on by restricting all variables of the constructed ILP to be non-negative.

$\mathcal{C}_{\ell,1}$ to $\mathcal{C}_{\ell,3}$ ensure that the currently most significant bit is assigned correctly. As the values of $y(\ell)$ are assigned from the most significant bit to the least significant bit, this ensures the correctness of the binary encoding. More specifically, we have the following property.

► **Observation 8.** Constraints $\mathcal{C}_{\ell,1}$ to $\mathcal{C}_{\ell,3}$ imply

$$y(\ell) \begin{cases} = 0, & \text{if } r(\ell + 1) < \gamma^{2^\ell} \\ \geq 1 & \text{if } r(\ell + 1) \geq \gamma^{2^\ell} \end{cases}$$

Proof. We show this by analyzing the cases separately:

Case 1: Assume $r(\ell + 1) < \gamma^{2^\ell}$. $\mathcal{C}_{\ell,2}$ states $y(\ell) \leq r(\ell + 1)/\gamma^{2^\ell} + 1/(\gamma^{2^\ell} + 1)$. With the assumption and the fact that $r(\ell + 1)$ is integer, we get $y(\ell) \leq (\gamma^{2^\ell} - 1)/\gamma^{2^\ell} + 1/(\gamma^{2^\ell} + 1)$. Now, $1/(\gamma^{2^\ell} + 1) < 1/\gamma^{2^\ell}$ implies $y(\ell) < \gamma^{2^\ell}/\gamma^{2^\ell} = 1$. Since $y(\ell)$ is also restricted to integer values we get $y(\ell) = 0$ with $\mathcal{C}_{\ell,1}$. Now, we only need to show that the other constraint is also fulfilled for $y(\ell) = 0$. For $\mathcal{C}_{\ell,3}$ we get $0 = y(\ell) \geq r(\ell + 1)/\gamma^{2^\ell} - (\gamma^{2^\ell} - 1)/\gamma^{2^\ell}$, since $r(\ell + 1) \leq \gamma^{2^\ell} - 1$ which completes this part of the proof.

Case 2: Assume $r(\ell + 1) \geq \gamma^{2^\ell}$. $\mathcal{C}_{\ell,3}$ states $y(\ell) \geq r(\ell + 1)/\gamma^{2^\ell} - (\gamma^{2^\ell} - 1)/\gamma^{2^\ell}$. With the assumption, we get $y(\ell) \geq 1 - (\gamma^{2^\ell} - 1)/\gamma^{2^\ell} > 0$. Since $y(\ell)$ is restricted to integer values we get $y(\ell) \geq 1$. Now, we only need to show that the other constraints are also fulfilled for $y(\ell)$. This clearly holds for $\mathcal{C}_{\ell,1}$. For $\mathcal{C}_{\ell,2}$ we get $1 \leq y(\ell) \leq r(\ell + 1)/\gamma^{2^\ell} + 1/(\gamma^{2^\ell} + 1)$, since $r(\ell + 1)/\gamma^{2^\ell} \geq 1$ and $1/(\gamma^{2^\ell} + 1) \leq 1$. This finalizes the proof. ◀

In particular, we have that the value of $y(\ell) = \lfloor r(\ell+1)/\gamma^{2^\ell} + 1/(\gamma^{2^\ell}+1) \rfloor$ is unique for given $r(\ell+1)$. This is due to the fact that the feasible interval (given by $\mathcal{C}_{\ell,2}$ and $\mathcal{C}_{\ell,3}$) is of size $1 - 1/\gamma^{2^\ell} \leq 1/(\gamma^{2^\ell}+1) + (\gamma^{2^\ell}-1)/\gamma^{2^\ell} < 1$. The second inequality implies that there exists at most one integer value in the interval. The first inequality in combination with the fact that $r(\ell+1)$ is integer, we also have that there must exist an integer value in the interval.

Now, we are able to show that the $x^{\text{bin}}(\ell), \ell \in [d-1]_0$ matches the binary encoding of i . $\mathcal{C}_{\ell,6}$ ensures that $x^{\text{bin}}(\ell) = 0$ if $y(\ell) = 0$, while $\mathcal{C}_{\ell,7}$ provides that $x^{\text{bin}}(\ell) > 0$ if $y(\ell) > 0$. Taken together with $\mathcal{C}_{\ell,5}$ and the fact that $x^{\text{bin}}(\ell)$ is integer, we observe the following.

► **Observation 9.** *Constraints $\mathcal{C}_{\ell,1}$ to $\mathcal{C}_{\ell,7}$ imply*

$$x^{\text{bin}}(\ell) = \begin{cases} 0, & \text{if } r(\ell+1) < \gamma^{2^\ell} \\ 1, & \text{if } r(\ell+1) \geq \gamma^{2^\ell} \end{cases}$$

Proof. We show this by again separately analyzing the cases:

Case 1: Assume $r(\ell+1) < \gamma^{2^\ell}$. With Observation 8, we know $y(\ell) = 0$. $\mathcal{C}_{\ell,6}$ now implies $x^{\text{bin}}(\ell) = 0$. This also fulfills $\mathcal{C}_{\ell,4}, \mathcal{C}_{\ell,5}$ and $\mathcal{C}_{\ell,7}$.

Case 2: Assume $r(\ell+1) \geq \gamma^{2^\ell}$. Observation 8 and $\mathcal{C}_{\ell,7}$ now imply $x^{\text{bin}}(\ell) \geq 1$. With $\mathcal{C}_{\ell,5}$ we get $x^{\text{bin}}(\ell) = 1$. This also fulfills $\mathcal{C}_{\ell,4}$ and $\mathcal{C}_{\ell,6}$ and finalizes the proof.

Note that this also upper bounds the value of $y(\ell)$. Since $x^{\text{bin}}(\ell) \leq 1$, $\mathcal{C}_{\ell,7}$ implies $y(\ell) \leq \gamma^{2^\ell} + 1$. ◀

And finally, $\mathcal{C}_{\ell,10}$ to $\mathcal{C}_{\ell,13}$ represent the remainder calculation and ensure that the constraints can be used recursively. More concretely, we have the following behavior.

► **Observation 10** (\approx). *Constraints $\mathcal{C}_{\ell,8}$ to $\mathcal{C}_{\ell,13}$ imply*

$$r(\ell) = \begin{cases} r(\ell+1), & \text{if } r(\ell+1) < \gamma^{2^\ell} \\ r(\ell+1)/\gamma^{2^\ell}, & \text{if } r(\ell+1) \geq \gamma^{2^\ell} \end{cases}$$

Also, for each $\ell \in [d]_0$, the remainder is bounded by $r(\ell) \leq \gamma^{2^\ell} - 1$.

Proof Outline. We prove this statement via case distinction and induction over ℓ . Due to space concerns the proof can be found in the full version. ◀

Now, we prove that if the Constraints 1 are feasible, then $r(d)$ is of the form $\gamma^i, i \in \mathbb{Z}_{\geq 1}$ and x^{bin} is the binary encoding of i . We also show that the constraints do not permit any other integer solutions.

► **Lemma 11.** *Let $d \in \mathbb{Z}_{\geq 1}$ and $\gamma \in \mathbb{Z}_{\geq 2}$. If there exists an $i \in \mathbb{Z}_{\geq 1}$ such that $r(d) = \gamma^i, i \in \mathbb{Z}_{\geq 1}$, with $\gamma^i \leq \gamma^{2^d-1}$, then x^{bin} is the binary encoding of i .*

Proof. Assume, there exists an $i \in \mathbb{Z}_{\geq 1}$ such that $r(d) = \gamma^i$ and $\gamma^i \leq \gamma^{2^d-1}$. We prove that x^{bin} is the binary encoding of i by induction over $\ell = d-1, \dots, 0$.

Base Case: By the assumption $r(d) = \gamma^i$, we have that $r(d)$ is correctly set.

Inductive Step: Let $\ell \in [d-1]$ and assume that $r(\ell)$ is the correct remainder in the ℓ -th iteration. Following the procedure described at the beginning of this section, we know that $x^{\text{bin}}(\ell-1)$ is required to be 1 if $r(\ell) \geq \gamma^{2^{\ell-1}}$ and 0 otherwise. This property holds due to Observations 8 and 9. By Observation 10, we know that the next remainder $r(\ell-1)$ is correctly set to $r(\ell)/\gamma^{2^{\ell-1}}$ if $r(\ell) \geq \gamma^{2^{\ell-1}}$ and to $r(\ell)$ otherwise. ◀

48:12 The Support of Bin Packing Is Exponential

Now, we know that x^{bin} is the binary encoding of i , if there exists $i \in \mathbb{Z}_{\geq 1}$ with $\gamma^i \leq \gamma^{2^d-1}$ and $r(d) = \gamma^i$. To finalize that the constraints fulfill the desired property, we need to show that the constraints are infeasible for any other value of $r(d)$.

► **Lemma 12.** *Let $d \in \mathbb{Z}_{\geq 1}$ and $\gamma \in \mathbb{Z}_{\geq 2}$. If $r(d) = \gamma^i$ for some $i \notin \mathbb{Z}_{\geq 1}$ holds, then the Constraints 1 are infeasible.*

Proof. Assume $r(d) = \gamma^i$ for some $i \notin \mathbb{Z}_{\geq 1}$. Note that we can formulate any integer number k in the form $\gamma^{\log k / \log \gamma}$. The proof is split into different cases. For each case, we lead the constraints to a contradiction.

Case 1: First, we show that the set of constraints is infeasible when i is negative. Let $d \in \mathbb{Z}_{\geq 1}$ and $\gamma \in \mathbb{Z}_{\geq 2}$. Set $j := -i$. By the assumption, we have $r(d) = \gamma^i = 1/\gamma^j$. This contradicts the constraint that $r(d)$ is required to be integer.

Case 2: We now show that the set of constraints is infeasible when i is not integer. Let $\gamma \in \mathbb{Z}_{\geq 2}$ and $i \notin \mathbb{Z}_{\geq 0}$. Again set $r(d) = \gamma^i$. We now show by induction that the exponent $p(\ell)$ of γ is not integer for all $\ell \in [d]$. This then leads to a contradiction. This contradiction appears either at the additional constraint $r(0) = 1$ or already at constraint $\mathcal{C}_{k,9}$ for some $k \in [d-1]_0$.

Base Case: By definition of i in this case, i is not integer and with the assumption $r(d) = \gamma^i = \gamma^{p(d)}$ we get $i = p(d)$ and therefore $p(d)$ is not integer. Note that the assumption that the value $\gamma^{p(d)}$ is assigned to the variable $r(d)$ implies that $\gamma^{p(d)}$ must be integer.

Inductive Step: Let $\ell \in [d]$ and assume $p(\ell)$ is not integer and $r(\ell) = \gamma^{p(\ell)}$ is integer. Now we can meet the following two cases. Either, $r(\ell) < \gamma^{2^{\ell-1}}$ or $r(\ell) \geq \gamma^{2^{\ell-1}}$. In the first case, Observation 10 gives $r(\ell-1) = r(\ell)$ which directly implies $p(\ell-1) = p(\ell)$. Thus, $p(\ell-1)$ is not integer and $r(\ell-1)$ is integer. In the second case, with Observation 10 we get $r(\ell-1) = r(\ell)/\gamma^{2^{\ell-1}} = \gamma^{p(\ell)}/\gamma^{2^{\ell-1}} = \gamma^{p(\ell)-2^{\ell-1}}$. Since $p(\ell)$ is not integer and $2^{\ell-1}$ is integer, we get that $p(\ell-1) = p(\ell) - 2^{\ell-1}$ is not integer. If now $r(\ell-1) = \gamma^{p(\ell-1)}$ is integer, we repeat the inductive step. However, if $r(\ell-1) = \gamma^{p(\ell-1)}$ is not integer, there is no feasible integer assignment for $r(\ell-1)$ as $\mathcal{C}_{\ell,9}$ is an equality constraint.

If $r(\ell)$ is integer for all $\ell \in [d]$, we get that $p(0)$ is not integer. This implies $r(0) = \gamma^{p(0)}$. This is a contradiction to $r(0) = 1$.

Case 3: The only case left is $\gamma^i = 1$, i.e., $i = 0$. Again, by the assumption, we get $r(d) = \gamma^i = 1$ which directly contradicts $\mathcal{C}_{\alpha-3}$ which states $r(d) \geq 2$. ◀

Thus, we have shown that the Constraints 1 are feasible if and only if variable $r(d)$ takes a value of the form γ^i for given $\gamma \in \mathbb{Z}_{\geq 2}$ and positive integer i (Lemma 12). Additionally, x^{bin} matches the binary encoding of i , i.e., it holds that $i = \sum_{\ell=0}^{d-1} 2^\ell x^{\text{bin}}(\ell)$ (Lemma 11).

The attentive reader might have noticed that in Example 7 the values of $y(\ell)$ and $z(\ell)$ match for all values of i and ℓ . In the full version, we show that we can not simply replace $z(\ell)$ with $y(\ell)$ in $\mathcal{C}_{\ell,9}$.

4.3 Aggregation of the Constraints

Before we can aggregate the constraints into a single knapsack constraint, we need to transform the inequalities into equations such that we achieve an ILP of the form considered in Section 3. We can do that by introducing slack variables. Also, we omit the lower bounds $\mathcal{C}_{\ell,1}, \mathcal{C}_{\ell,4}, \mathcal{C}_{\ell,8}$ and $\mathcal{C}_{\ell,10}$ by restricting all variables to only take non-negative values. This results in the following new set of constraints:

► Constraints 2.

$$\begin{aligned}
r(0) &= 1 \\
(\gamma^{2^{\ell+1}} + \gamma^{2^\ell})y(\ell) - (\gamma^{2^\ell} + 1)r(\ell + 1) + s_1(\ell) &= \gamma^{2^\ell} & \forall \ell \in [d-1]_0 \\
-\gamma^{2^\ell}y(\ell) + r(\ell + 1) + s_2(\ell) &= \gamma^{2^\ell} - 1 & \forall \ell \in [d-1]_0 \\
x^{\text{bin}}(\ell) + s_3(\ell) &= 1 & \forall \ell \in [d-1]_0 \\
x^{\text{bin}}(\ell) - y(\ell) + s_4(\ell) &= 0 & \forall \ell \in [d-1]_0 \\
y(\ell) - (\gamma^{2^\ell} + 1)x^{\text{bin}}(\ell) + s_5(\ell) &= 0 & \forall \ell \in [d-1]_0 \\
(\gamma^{2^\ell} - 1)z(\ell) + r(\ell) - r(\ell + 1) &= 0 & \forall \ell \in [d-1]_0 \\
\gamma^{2^\ell}x^{\text{bin}}(\ell) - z(\ell) + r(\ell) + s_7(\ell) &= \gamma^{2^\ell} & \forall \ell \in [d-1]_0 \\
-\gamma^{2^\ell}x^{\text{bin}}(\ell) + z(\ell) + s_8(\ell) &= 0 & \forall \ell \in [d-1]_0 \\
z(\ell) - r(\ell) + s_9(\ell) &= 0 & \forall \ell \in [d-1]_0 \\
r(d) - s_{\alpha-3} &= 2 \\
r(d) + s_{\alpha-2} &= \gamma^{2^d-1}
\end{aligned}$$

Now, we add the upper bounds of the variables. With Observation 10, we have $r(\ell) \leq \gamma^{2^\ell} - 1$. In combination with Observation 8, we get $y(\ell) \leq \lfloor (\gamma^{2^{\ell+1}} - 1)/\gamma^{2^\ell} + 1/(\gamma^{2^\ell} + 1) \rfloor \leq \gamma^{2^\ell}$. $x^{\text{bin}}(\ell)$ is bounded by 1 due to $\mathcal{C}_{\ell,5}$ and it holds that $z(\ell) \leq \min(\gamma^{2^\ell}, \gamma^{2^\ell} - 1) = \gamma^{2^\ell} - 1$ by $\mathcal{C}_{\ell,12}$ and $\mathcal{C}_{\ell,13}$. The slack variables are bounded by the coefficients, the variables and the right-hand-side of their constraint. For instance, since $\gamma^{2^\ell}x^{\text{bin}}(\ell) - z(\ell) + r(\ell) + s_7(\ell) = \gamma^{2^\ell}$, we have $s_7(\ell) = \gamma^{2^\ell} - \gamma^{2^\ell}x^{\text{bin}}(\ell) + z(\ell) - r(\ell)$. With $x^{\text{bin}}(\ell) \geq 0$, $r(\ell) \geq 0$ and the bound on $z(\ell)$, this implies $s_7(\ell) \leq \gamma^{2^\ell} + \gamma^{2^\ell} - 1 = 2\gamma^{2^\ell} - 1$.

Now, as done in Section 3, we define U to be the sum of the upper bounds of all variables (including slacks) and introduce the constraint which restricts the sum of all variables by U . Let $v := \sum_{\ell=0}^{d-1} (y(\ell) + z(\ell) + r(\ell) + \sum_{j=1}^9 (s_j(\ell))) + r(d) + s_{\alpha-3} + s_{\alpha-2}$ be the sum of all variables. Thus, our additional constraint is $v + s_\alpha = U$. Setting $M := 2\gamma^{2^d} \cdot U + \gamma^{2^d-1} + 2\gamma^{2^d} + 2$ and multiplying the ILP with the vector $(1, M, M^2, \dots, M^{\alpha-1})$ results in an ILP of the form (3) which is equivalent to the Constraints 2.

Lemma 5 now implies that vector $\text{sol} = (x^{\text{bin}}, y, z, r, s)^T \in \mathbb{Z}_{\geq 0}^{12d+4}$ is a feasible solution to Constraints 2 if and only if sol is a solution to

$$\begin{aligned}
r(0) + \sum_{\ell=0}^{d-1} \Big(& M^{9\ell+1}((\gamma^{2^{\ell+1}} + \gamma^{2^\ell})y(\ell) - (\gamma^{2^\ell} + 1)r(\ell + 1) + s_1(\ell)) \\
& + M^{9\ell+2}(-\gamma^{2^\ell}y(\ell) + r(\ell + 1) + s_2(\ell)) \\
& + M^{9\ell+3}(x^{\text{bin}}(\ell) + s_3(\ell)) \\
& + M^{9\ell+4}(x^{\text{bin}}(\ell) - y(\ell) + s_4(\ell)) \\
& + M^{9\ell+5}(y(\ell) - (\gamma^{2^\ell} + 1)x^{\text{bin}}(\ell) + s_5(\ell)) \\
& + M^{9\ell+6}((\gamma^{2^\ell} - 1)z(\ell) + r(\ell) - r(\ell + 1)) \\
& + M^{9\ell+7}(\gamma^{2^\ell}x^{\text{bin}}(\ell) - z(\ell) + r(\ell) + s_7(\ell)) \\
& + M^{9\ell+8}(-\gamma^{2^\ell}x^{\text{bin}}(\ell) + z(\ell) + s_8(\ell)) \\
& + M^{9\ell+9}(z(\ell) - r(\ell) + s_9(\ell)) \Big) \\
& + M^{\alpha-3}(r(d) - s_{\alpha-3}) \\
& + M^{\alpha-2}(r(d) + s_{\alpha-2}) \\
& + M^{\alpha-1}(v + s_\alpha) \\
& = 1 + \sum_{\ell=1}^{d-1} \left(M^{9\ell+1}\gamma^{2^\ell} + M^{9\ell+2}(\gamma^{2^\ell} - 1) + M^{9\ell+3} + M^{9\ell+7}\gamma^{2^\ell} \right) \\
& + M^{\alpha-3} \cdot 2 + M^{\alpha-2}\gamma^{2^d-1} + M^{\alpha-1}U.
\end{aligned} \tag{7}$$

This equation can be transformed into an unbounded knapsack constraint by grouping coefficients such that each variable only occurs once. These coefficients then define the item size vector s . For example, the sizes of the item $x^{\text{bin}}(\ell)$, $\ell \in [d-1]_0$ is $s_{x^{\text{bin}}(\ell)} = M^{9\ell+3} + M^{9\ell+4} + M^{9\ell+7} \cdot \gamma^{2^\ell} - M^{9\ell+8} \cdot \gamma^{2^\ell}$. To complete the transformation, set the lower bound of each variable to 0 and exchange the equality into \leq , i.e., the sum of item sizes and multiplicities is at most the right hand side. Denote the polytope as $P = \{x \in \mathbb{R}^d | x_i, s^T x \leq C\}$. All feasible configurations are then integer points in P , i.e., $p \in P \cap \mathbb{Z}^d$.

4.4 Main proof

With this, we are ready to prove our main theorem.

► **Theorem 1.** *There exists a high-multiplicity BIN PACKING instance of dimension d such that the solution-vector x (a vector of configurations) contains at least $2^{\Omega(d)}$ non-zero entries, i.e., $|\text{supp}(x)| \in 2^{\Omega(d)}$.*

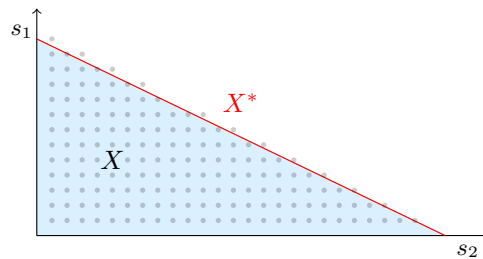
Proof. Let $d' \in \mathbb{Z}_{\geq 0}$. The dimension of the BIN PACKING instance has dimension d , i.e., number of different item sizes, $d = 12d' + 4$. Denote the capacity of a single bin as C and set the sizes according to (7), and denote them as the size vector s . Set C to be the right hand side of the knapsack constraint (7). Define the knapsack polytope $P = \{x \in \mathbb{R}^d | x_i, s^T x \leq C\}$. Let X be the set of all solutions in $P \cap \mathbb{Z}^d$, i.e., with total size less than or equal to C . Define $X^* = \{x \in X | s^T x = A\}$ to be the set of all solutions with size exactly C . See Figure 3 for an illustration. Thus, X is the set of all feasible configurations. By the definition of the knapsack constraint and Lemma 12, we know that $|X^*| = 2^{d'} - 1 =: k$, as (7) has exactly one solution for every possible combination of $x^{\text{bin}}(\ell)$, $\ell \in [d-1]_0$ with $x^{\text{bin}}(\ell) \in \{0, 1\}$. Define the total amount of items y as the sum of all configurations, i.e., $y = \sum_{x \in X^*} x$. Thus, the total size of all items to be placed is $y^T \cdot s = kC$.

Clearly, the optimal solution to this BIN PACKING instance uses exactly k bins. One possible solution is to take each configuration in X^* exactly once. In the following we show that this is the only solution using elements in X and using exactly k bins. Now, define the projection $\pi : x \rightarrow (x^{\text{bin}}(0), \dots, x^{\text{bin}}(d'-1), r(d'))$. This projection reduces every configuration

to just the binary encoding and the value $r(d')$. It holds that $\pi(y) = \begin{pmatrix} 2^{d-1} \\ \vdots \\ 2^{d-1} \\ \sum_{i=1}^k (4k)^i \end{pmatrix}$. Recall

that we set $\gamma = 4k$. We can not consider configurations in $X \setminus X^*$ as they have size $< C$. As the total size of the right hand side is exactly kC taking one of these configurations while still using k bins would require another bin to be filled $> C$, which is infeasible due to (7). Thus, there are no other feasible configurations, as only those of the form (after applying π) exactly like in Lemma 6, i.e., those in X^* , have size exactly C . Thus, when applying the projection π , we have met the conditions of applying Lemma 6 and know that the only optimal solution takes each of the k vectors exactly once.

Therefore, after removing the projection, the only optimal solution to the BIN PACKING instance uses every configuration in X^* exactly once, and therefore we have $|\text{supp}(x)| = k = 2^{d'} - 1$. As we have $d = 12d' + 4$ we have $|\text{supp}(x)| \in 2^{\Omega(d)}$. ◀



■ **Figure 3** A 2-dimensional example of the knapsack constraint with item types s_1, s_2 . The shaded area denotes the knapsack polytope P . The points resemble integer points and show all feasible configurations. Integer points in the shaded region are the configurations in the set X . Integer points on the red line form the set X^* of configurations. Only configurations in X^* fill the knapsack constraint with equality.

References

- 1 Iskander Aliev, Jesús A. De Loera, Friedrich Eisenbrand, Timm Oertel, and Robert Weismantel. The support of integer optimal solutions. *SIAM J. Optim.*, 28(3):2152–2157, 2018. doi:10.1137/17M1162792.
- 2 Iskander Aliev, Jesús A. De Loera, Timm Oertel, and Christopher O’Neill. Sparse solutions of linear diophantine equations. *SIAM J. Appl. Algebra Geom.*, 1(1):239–253, 2017. doi:10.1137/16M1083876.
- 3 Sebastian Berndt, Hauke Brinkop, Klaus Jansen, Matthias Mnich, and Tobias Stamm. New support size bounds for integer programming, applied to makespan minimization on uniformly related machines. In *ISAAC*, volume 283 of *LIPICs*, pages 13:1–13:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICs.ISAAC.2023.13.
- 4 Gordon H. Bradley. Transformation of integer programs to knapsack problems. *Discrete Mathematics*, 1(1):29–45, 1971. doi:10.1016/0012-365X(71)90005-7.
- 5 Gordon H. Bradley, Peter L. Hammer, and Laurence A. Wolsey. Coefficient reduction for inequalities in 0-1 variables. *Math. Program.*, 7(1):263–282, 1974. doi:10.1007/BF01585527.
- 6 Václav Chvátal and Peter Hammer. Aggregation of inequalities in integer programming. *Ann. Discrete Math*, 1:145–162, 1977.
- 7 Friedrich Eisenbrand, Christoph Hunkenschröder, Kim-Manuel Klein, Martin Koutecký, Asaf Levin, and Shmuel Onn. Reducibility bounds of objective functions over the integers. *Oper. Res. Lett.*, 51(6):595–598, 2023. doi:10.1016/J.ORL.2023.10.001.
- 8 Friedrich Eisenbrand and Gennady Shmonin. Carathéodory bounds for integer cones. *Oper. Res. Lett.*, 34(5):564–568, 2006. doi:10.1016/J.ORL.2005.09.008.
- 9 SE Elmaghraby and MK Wig. On the treatment of stock cutting problems as diophantine programs. *Operations Research Report*, 61, 1970.
- 10 András Frank and Éva Tardos. An application of simultaneous diophantine approximation in combinatorial optimization. *Comb.*, 7(1):49–65, 1987. doi:10.1007/BF02579200.
- 11 Fred W. Glover and Djangir A. Babayev. New results for aggregating integer-valued equations. *Ann. Oper. Res.*, 58(3):227–242, 1995. doi:10.1007/BF02032133.
- 12 Fred W. Glover and Robert E. D. Wolsey. Aggregating diophantine equations. *Z. Oper. Research*, 16(1):1–10, 1972. doi:10.1007/BF01917186.
- 13 Michel X. Goemans and Thomas Rothvoss. Polynomiality for bin packing with a constant number of item types. *J. ACM*, 67(6):38:1–38:21, 2020. doi:10.1145/3421750.
- 14 A. C. Hayes and David G. Larman. The vertices of the knapsack polytope. *Discret. Appl. Math.*, 6(2):135–138, 1983. doi:10.1016/0166-218X(83)90067-7.

- 15 Klaus Jansen. An EPTAS for scheduling jobs on uniform processors: Using an MILP relaxation with a constant number of integral variables. *SIAM J. Discret. Math.*, 24(2):457–485, 2010. doi:10.1137/090749451.
- 16 Klaus Jansen, Kai Kahler, and Esther Zwanger. Exact and approximate high-multiplicity scheduling on identical machines. *CoRR*, abs/2404.17274, 2024. doi:10.48550/arXiv.2404.17274.
- 17 Klaus Jansen and Kim-Manuel Klein. A robust AFPTAS for online bin packing with polynomial migration. *SIAM J. Discret. Math.*, 33(4):2062–2091, 2019. doi:10.1137/17M1122529.
- 18 Klaus Jansen and Kim-Manuel Klein. About the structure of the integer cone and its application to bin packing. *Math. Oper. Res.*, 45(4):1498–1511, 2020. doi:10.1287/MOOR.2019.1040.
- 19 Klaus Jansen, Kim-Manuel Klein, and José Verschae. Closing the gap for makespan scheduling via sparsification techniques. *Math. Oper. Res.*, 45(4):1371–1392, 2020. doi:10.1287/MOOR.2019.1036.
- 20 Klaus Jansen and Roberto Solis-Oba. A polynomial time $OPT + 1$ algorithm for the cutting stock problem with a constant number of object lengths. *Math. Oper. Res.*, 36(4):743–753, 2011. doi:10.1287/MOOR.1110.0515.
- 21 Kenneth E Kendall and Stanley Zionts. Solving integer programming problems by aggregating constraints. *Operations Research*, 25(2):346–351, 1977.
- 22 Manfred W. Padberg. Equivalent knapsack-type formulations of bounded integer linear programs: An alternative approach. *Naval Research Logistics Quarterly*, 19(4):699–708, 1972. doi:10.1002/nav.3800190410.
- 23 Pierre-Louis Poirion. Optimal constraints aggregation method for ILP. *Discret. Appl. Math.*, 262:148–157, 2019. doi:10.1016/J.DAM.2019.02.014.
- 24 I.G. Rosenberg. Aggregation of equations in integer programming. *Discrete Mathematics*, 10(2):325–341, 1974. doi:10.1016/0012-365X(74)90126-5.