Polynomiality for Bin Packing with a Constant Number of Item Types*

Michel X. Goemans† Thomas Rothvoß‡

Abstract

We consider the bin packing problem with d different item sizes s_i and item multiplicities a_i, where all numbers are given in binary encoding. This problem formulation is also known as the *1-dimensional cutting stock problem*.

In this work, we provide an algorithm which, for constant d, solves bin packing in polynomial time. This was an open problem for all $d \geq 3$.

In fact, for constant d our algorithm solves the following problem in polynomial time: given two d-dimensional polytopes P and Q, find the smallest number of integer points in P whose sum lies in Q.

Our approach also applies to *high multiplicity* scheduling problems in which the number of copies of each job type is given in binary encoding and each type comes with certain parameters such as release dates, processing times and deadlines. We show that a variety of high multiplicity scheduling problems can be solved in polynomial time if the number of job types is constant.

1 Introduction

Let (s, a) be an instance for *bin packing* with *item sizes* $s_1, \ldots, s_d \in [0, 1]$ and a vector $a \in \mathbb{Z}^d_{\geq 0}$ of *item multiplicities*. In other words, our instance contains a_i many copies of an item of size s_i. In the following we assume that s_i is given as a rational number and Δ is the largest number appearing in the denominator of s_i or the multiplicities a_i. Let $\mathcal{P} := \{x \in \mathbb{Z}^d_{\geq 0} \mid s^T x \leq 1\}$. Now the goal is to select a minimum number of vectors from $\mathcal{P}$ that sum up to a, i.e.

$$\min \Big\{ \mathbf{1}^T \lambda \mid \sum_{x \in \mathcal{P}} \lambda_x \cdot x = a; \;\; \lambda \in \mathbb{Z}^{\mathcal{P}}_{\geq 0} \Big\} \tag{1.1}$$

where λ_x is the *weight* that is given to $x \in \mathcal{P}$. This problem is also known as the *(1-dimensional) cutting stock problem* and its study goes back to the classical paper by Gilmore and Gomory [GG61]. Note that even for fixed dimension d, the problem is that both, the number of points $|\mathcal{P}|$ and the weights λ_x will be exponentially large. Let OPT and OPT_f be the optimum integral and fractional solution to (1.1). As bin packing for general d is strongly **NP**-hard [Joh92], we are particularly interested in the complexity of bin packing if d is constant. For $d = 2$ it is true that $OPT = \lceil OPT_f \rceil$ and it suffices to compute and round an optimum fractional solution [MSS97]. However, for $d \geq 3$, one might have $OPT > \lceil OPT_f \rceil$. Still, [FA05] generalized the argument of [MSS97] to find a solution with at most $d - 2$ bins more than the optimum in polynomial time.

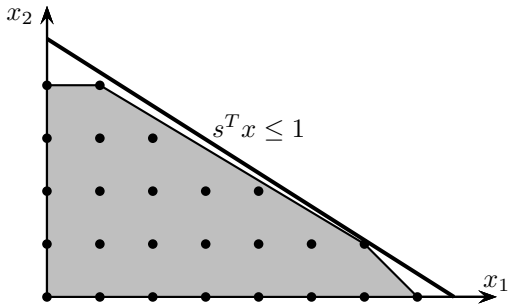


Figure 1: Knapsack polytope for $s = (0.13, 0.205)$.

The best polynomial time algorithm previously known for constant $d \geq 3$ is an $OPT + 1$ approximation algorithm by Jansen and Solis-Oba [JSO10] which runs in time $2^{2^{O(d)}} \cdot (\log \Delta)^{O(1)}$. Their algorithm is based on the following insights: (1) If all items are small, say $s_i \leq \frac{1}{2d}$, then the integrality gap is at most one[1]. (2) If all items have constant size, then one can guess the points used in the optimum solution. It turns out that for arbitrary instances both approaches can be combined for an $OPT + 1$ algorithm. However, to find an optimum solution, we cannot allow any error and a fundamentally different approach is needed.

Note that for general d, the recent algorithm of the 2nd author provides solutions of cost at most $OPT + O(\log d \cdot \log \log d)$ [Rot13], improving the classical Karmarkar-Karp algorithm with a guarantee of

*Research supported in part by ONR grant N00014-11-1-0053 and by NSF contract 1115849.

†MIT, Email: goemans@math.mit.edu.

‡MIT, Email: rothvoss@math.mit.edu.

[1] Compute a basic solution λ to the LP and buy $\lfloor \lambda_x \rfloor$ times point x. Then assign the items in the remaining instance greedily.

$OPT + O(\log^2 d)$ [KK82]. Both algorithms run in time polynomial in $\sum_{i=1}^{d} a_i$ and thus count in our setting as pseudopolynomial. In fact, those algorithms can still be cast as asymptotic FPTAS.

Bin packing and more generally the cutting stock problem belong to a family of problems that consist of selecting integer points in a polytope with multiplicities. In fact, several scheduling problems fall into this framework as well, where the polytope describes the set of jobs that are admissible on a machine under various constraints.

We give some notation needed throughout the paper. For a set $X \subseteq \mathbb{R}^d$, we define the spanned *cone* as $\text{cone}(X) = \{\sum_{x\in X} \lambda_x x \mid \lambda_x \geq 0 \; \forall x \in X\}$ and the *integer cone* as $\text{int.cone}(X) := \{\sum_{x\in X} \lambda_x x \mid \lambda_x \in \mathbb{Z}_{\geq 0} \; \forall x \in X\}$. For a polytope $P = \{x \in \mathbb{R}^d \mid Ax \leq b\}$, we define $\text{enc}(P)$ as the number of bits that it takes to write down the inequalities defining P. Note that $\text{enc}(P)$ is polynomially related to $\max\{m, \log \Delta\}$ where m is the number of inequalities and Δ is the largest number appearing in an integral inequality representation of P.

2 Our contributions

In this paper, we resolve the question of whether bin packing with a fixed number of item types can be solved in polynomial time.

THEOREM 2.1. *For any Bin Packing instance (s, a) with $s \in [0,1]^d$ and $a \in \mathbb{Z}^d_{\geq 0}$, an optimum integral solution can be computed in time $(\log \Delta)^{2^{O(d)}}$ where Δ is the largest integer appearing in a denominator s_i or in a multiplicity a_i.*

This answers an open question posed by McCormick, Smallwood and Spieksma [MSS97] as well as by Eisenbrand and Shmonin [ES06]. In fact, the first paper even conjectured this problem to be **NP**-hard for $d = 3$. Moreover the polynomial solvability for general d was called a "*hard open problem*" by Filippi [Fil07].

In fact, we derive Theorem 2.1 via the following general theorem for finding conic integer combinations in fixed dimension.

THEOREM 2.2. *Given polytopes $P, Q \subseteq \mathbb{R}^d$, one can find a vector $y \in$ int.cone$(P \cap \mathbb{Z}^d) \cap Q$ and a vector $\lambda \in \mathbb{Z}^{P\cap\mathbb{Z}^d}_{\geq 0}$ such that $y = \sum_{x\in P\cap\mathbb{Z}^d} \lambda_x x$ in time $\text{enc}(P)^{2^{O(d)}} \cdot \text{enc}(Q)^{O(1)}$, or decide that no such y exists. Moreover, the support of λ is always bounded by 2^{2d+1}.*

In fact, by choosing $P = \{\binom{x}{1} \in \mathbb{R}^{d+1}_{\geq 0} \mid s^T x \leq 1\}$ and $Q = \{a\} \times [0, b]$, we can decide in polynomial time, whether b bins suffice. Theorem 2.1 then follows using binary search.

For the sake of a simple presentation, we assume that P is a bounded polytope. Our main insight to prove Theorem 2.2 lies in the following structure theorem which says that, for fixed d, there is a pre-computable polynomial size set $X \subseteq P \cap \mathbb{Z}^d$ of special vectors that are *independent* of the target polytope Q with the property that, for any $y \in$ int.cone$(P \cap \mathbb{Z}^d) \cap Q$, there is always a conic integer combination that has all but a constant amount of weight on a constant number of vectors in X.

THEOREM 2.3. (STRUCTURE THEOREM) *Let $P = \{x \in \mathbb{R}^d \mid Ax \leq b\}$ be a polytope with $A \in \mathbb{Z}^{m\times d}, b \in \mathbb{Z}^m$ such that all coefficients are bounded by Δ in absolute value. Then there exists a set $X \subseteq P \cap \mathbb{Z}^d$ of size $|X| \leq N := m^d d^{O(d)} (\log \Delta)^d$ that can be computed in time $N^{O(1)}$ with the following property: For any vector $a \in$ int.cone$(P \cap \mathbb{Z}^d)$ there exists an integral vector $\lambda \in \mathbb{Z}^{P\cap\mathbb{Z}^d}_{\geq 0}$ such that $\sum_{x\in P\cap\mathbb{Z}^d} \lambda_x x = a$ and*

(1) $\lambda_x \in \{0, 1\} \;\; \forall x \in (P \cap \mathbb{Z}^d) \backslash X$

(2) $|\text{supp}(\lambda) \cap X| \leq 2^{2d}$

(3) $|\text{supp}(\lambda) \backslash X| \leq 2^{2d}$.

With this structure theorem one can obtain Theorem 2.2 simply by computing X, guessing $\text{supp}(\lambda) \cap X$ and finding the corresponding values of λ and the vectors in $\text{supp}(\lambda) \backslash X$ with an integer program with a constant number of variables.

Bin packing can also be considered as a scheduling problem where the processing times correspond to the item sizes and the number of machines should be minimized, given a bound on the makespan. A variety of scheduling problems in the so-called high multiplicity setting can also be tackled using Theorem 2.2. Some of these scheduling applications are described in Section 7. For example we can solve in polynomial time the high multiplicity variant of minimizing the makespan for unrelated machines with machine-dependent release dates for a fixed number of job types and machine types.

3 Preliminaries

In this section we are going to review some known tools that we are going to use in our algorithm. The first one is Lenstra's well known algorithm for integer progamming, that runs in polynomial time as long as d is fixed[2].

THEOREM 3.1. (LENSTRA [LEN83], KANNAN [KAN87]) *Given $A \in \mathbb{Z}^{m\times d}$ and $b \in \mathbb{Z}^m$ with $\Delta := \max\{\|A\|_\infty, \|b\|_\infty\}$. Then one can find an $x \in \mathbb{Z}^d$*

[2]Here, the original dependence of [Len83] was $2^{O(d^3)}$ which was then improved by Kannan [Kan87] to $d^{O(d)}$.

with $Ax \leq b$ (or decide that none exists) in time $d^{O(d)} \cdot m^{O(1)} \cdot (\log \Delta)^{O(1)}$.

For a polytope $P \subseteq \mathbb{R}^d$, the *integral hull* is the convex hull of the integral points, abbreviated with $P_I := \text{conv}(P \cap \mathbb{Z}^d)$ and the extreme points of P are denoted by $\text{vert}(P)$. If we consider a low dimensional polytope P, then P can indeed contain an exponential number of integral points — but only few of those can be extreme points of P_I.

THEOREM 3.2. (COOK ET AL. [CHKM92, HAR88]) *Consider any polytope $P = \{x \in \mathbb{R}^d \mid Ax \leq b\}$ with m constraints with $\Delta := \max\{\|A\|_\infty, \|b\|_\infty\} \geq 2$. Then $P_I = conv(P \cap \mathbb{Z}^d)$ has at most $m^d \cdot (O(\log \Delta))^d$ many extreme points. In fact a list of extreme points can be computed in time $d^{O(d)}(m \cdot \log(\Delta))^{O(d)}$.*

We will later refer to the coefficients λ_x as the *weight* given to x. For a vector $a \in \text{cone}(X)$ we know by *Carathéodory's Theorem* that there is always a corresponding vector $\lambda \geq \mathbf{0}$ with at most d non-zero entries and $a = \sum_{x \in X} \lambda_x x$. One may wonder how many points x are actually needed to generate some point in the integer cone. In fact, at least under the additional assumption that X is the set of integral points in a convex set, one can show that 2^d points suffice[3]. The arguments are crucial for our proofs, so we replicate the proof of [ES06] to be selfcontained.

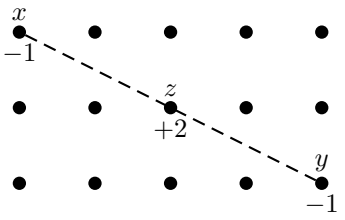
LEMMA 3.1. (EISENBRAND AND SHMONIN [ES06]) *For any polytope $P \subseteq \mathbb{R}^d$ and any integral vector $\lambda \in \mathbb{Z}_{\geq 0}^{P \cap \mathbb{Z}^d}$ there exists a $\mu \in \mathbb{Z}_{\geq 0}^{P \cap \mathbb{Z}^d}$ such that $|supp(\mu)| \leq 2^d$ and $\sum_x \mu_x x = \sum_x \lambda_x x$. Moreover $supp(\mu) \subseteq conv(supp(\lambda))$.*

Proof. For the sake of simplicity we can replace the original P with $P := \text{conv}(x \mid \lambda_x > 0)$ without changing the claim. Let $f : \mathbb{R}^d \to \mathbb{R}$ be any strictly convex function, i.e. in particular we will use that

$$f(\tfrac{1}{2}x + \tfrac{1}{2}y) < \frac{1}{2}(f(x) + f(y)).$$

For example $f(x) = \|(1, x)\|_2$ does the job. Let $(\mu_x)_{x \in P \cap \mathbb{Z}^d}$ be an integral vector with $\sum_{x \in P \cap \mathbb{Z}^d} \lambda_x x = \sum_{x \in P \cap \mathbb{Z}^d} \mu_x x$ that minimizes the potential function $\sum_{x \in P \cap \mathbb{Z}^d} \mu_x \cdot f(x)$ (note that there is at least one such solution, namely λ). In other words, we somewhat prefer points that are more in the "center" of the polytope. We claim that indeed $|\text{supp}(\mu)| \leq 2^d$.

For the sake of contradiction suppose that $|\text{supp}(\mu)| > 2^d$. Then there must be two points x, y with $\mu_x > 0$ and $\mu_y > 0$ that have the same *parity*, meaning that $x_i \equiv y_i \mod 2$ for all $i = 1, \ldots, d$. Then $z := \frac{1}{2}(x + y)$ is an integral vector and $z \in P$. Now we remove one unit of weight from both x and y and add 2 units to z.



This gives us another feasible vector μ'. But the change in the potential function is $+2f(z) - f(x) - f(y) < 0$ by strict convexity of f, contradicting the minimality of μ. □

In fact, the bound is tight up to a constant factor. As it seems that this has not been observed in the literature before, we describe a construction in Section 8 where a support of size 2^{d-1} is actually needed.

A family of versatile and well-behaved polytopes is those of *parallelepipeds*. Recall that

$$\Pi = \left\{ v_0 + \sum_{i=1}^{k} \mu_i v_i \mid -1 \leq \mu_i \leq 1 \; \forall i = 1, \ldots, k \right\}$$

is a *parallelepiped* with *center* $v_0 \in \mathbb{R}^d$ and *directions* $v_1, \ldots, v_k \in \mathbb{R}^d$. Usually one requires that the directions are linearly independent, that means $k \leq d$ and Π is k-dimensional. We say that the parallelepiped is *integral* if all its 2^k many vertices are integral.

4 Proof of the structure theorem

In this section we are going to prove the structure theorem. The proof outline is as follows: we can show that the integral points in a polytope P can be covered with polynomially many integral parallelepipeds. The choice for X is then simply the set of vertices of those parallelepipeds. Now consider any vector a which is a conic integer combination of points in P. Then by Lemma 3.1 we can assume that a is combined by using only a constant number of points in $P \cap \mathbb{Z}^d$. Consider such a point x^* and say it is used λ^* times. We will show that the weight λ^* can be almost entirely redistributed to the vertices of one of the parallelepipeds containing x^*.

Let us make these arguments more formal. We begin by showing that all the integer points in a polytope P can indeed be covered with polynomially many integral parallelepipeds as visualized in Figure 2.

[3]For arbitrary $X \subseteq \mathbb{Z}^d$, one can show that a support of at most $O(d \log(d\Delta))$ suffices, where Δ is the largest coefficient in a vector in X [ES06].

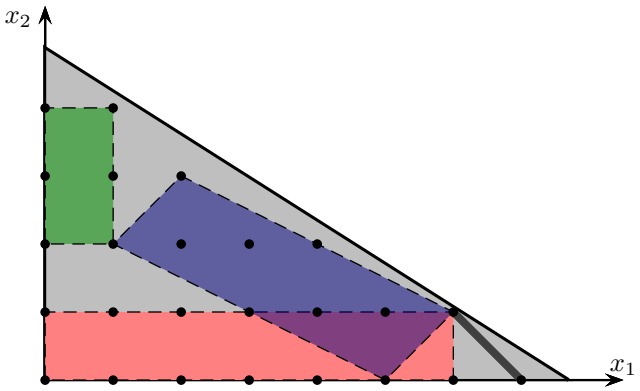


Figure 2: Covering the integer points of a polytope with integral parallelepipeds.

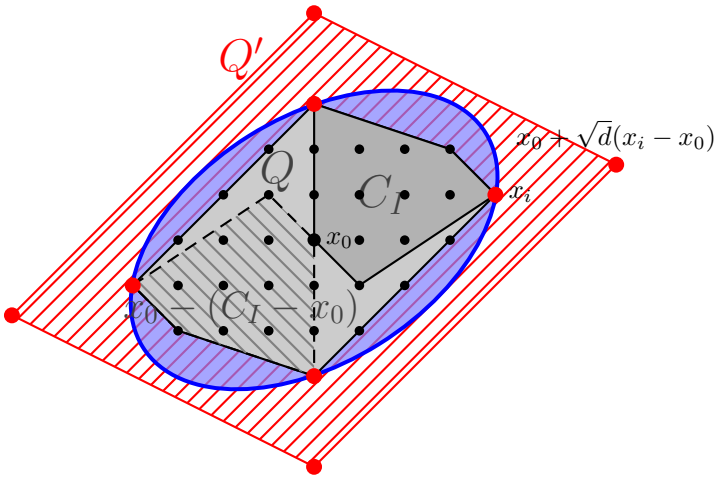


Figure 3: Visualization of covering the integer points in a cell C_I: Start by obtaining the symmetric closure Q. Then compute the contact points of a minimum volume ellipsoid containing Q. Scale those points with $\sqrt{d}$ to obtain a polytope Q' with only $O(d^2)$ vertices containing C_I. Then extend a triangulation of Q' to $d^{O(d)}$ many parallelepipeds.

LEMMA 4.1. *Let $P = \{x \in \mathbb{R}^d \mid Ax \le b\}$ be a polytope described by m inequalities with integral coefficients of absolute value at most Δ. Then there exists a set $\mathbf{\Pi}$ of at most $|\mathbf{\Pi}| \le N := m^d d^{O(d)} (\log \Delta)^d$ many integral parallelepipeds such that*

$$P \cap \mathbb{Z}^d \subseteq \bigcup_{\Pi \in \mathbf{\Pi}} \Pi \subseteq P.$$

Moreover the set $\mathbf{\Pi}$ can be computed in time $N^{O(1)}$.

Proof. First of all, remember that every point $x \in P$ has $\|x\|_\infty \le d! \cdot \Delta^d$ and hence $|A_i x - b_i| \le (d+1)\Delta \cdot d! \cdot \Delta^d \le (d+1)! \cdot \Delta^{d+1}$. We want to partition the interval $[0, (d+1)! \cdot \Delta^{d+1}]$ into smaller intervals $[\alpha_j, \alpha_{j+1}]$ such that for any integer values $p, q \in [\alpha_j, \alpha_{j+1}] \cap \mathbb{Z}$ one has $\frac{p}{q} \le 1 + \frac{1}{d^2}$. For this we can choose $\alpha_j := (1 + \frac{1}{d^2})^{j-2}$ for $j = 1, \dots, K$ and $\alpha_0 := 0$. It is not difficult to see that $K \le O(d^3(\log \Delta + \log d))$ such intervals suffice.

Our next step is to partition P into *cells* such that points in the same cell have roughly the same slacks for all the constraints. For each sequence $j_1, \dots, j_m \in \{1, \dots, K\}$ we define a cell $C = C(j_1, \dots, j_m)$ as

$$\left\{x \in \mathbb{R}^d \mid \alpha_{j_i} \le b_i - A_i x \le \alpha_{j_i+1} \ \forall i \in [m]\right\}.$$

In other words, we partition the polytope P using at most $M := m \cdot K$ many hyperplanes. By a perturbation argument our number of non-empty cells is bounded by the number of full dimensional cells in a hyperplane arrangement with M hyperplanes. It is a well-known result that the latter quantity is at most $\binom{M}{0} + \dots + \binom{M}{d} \le m^d d^{O(d)} (\log \Delta)^d$, see e.g. Matousek [Mat02].

Fix one of those non-empty cells $C \subseteq P$. We will show that there are only $d^{O(d)}$ parallelepipeds necessary to cover the integer points of this cell (we assume that $C \cap \mathbb{Z}^d \ne \emptyset$, otherwise there is nothing to do). Next, fix any integral point $x_0 \in C \cap \mathbb{Z}^d$ and define a slightly larger polytope $Q := \mathrm{conv}(x_0 \pm (x_0 - x) \mid x \in C_I)$, see Figure 3. In words, Q is a *centrally symmetric* polytope with integral vertices containing C_I such that also the center x_0 is integral. The reason why we consider a symmetric polytope is the following classical theorem which is paraphrased from John:

THEOREM 4.1. (JOHN [JOH48]) *For any centrally symmetric polytope $\tilde{P} \subseteq \mathbb{R}^d$, there are $k \le \frac{1}{2}d(d+3)$ many extreme points $x_1, \dots, x_k \in vert(\tilde{P})$ such that $\tilde{P} \subseteq conv(\sqrt{d} \cdot x_j \mid j \in [k])$.*

The original statement says that there is in fact an ellipsoid E with center $\mathbf{0}$ with $\frac{1}{\sqrt{d}}E \subseteq \tilde{P} \subseteq E$. But additionally John's Theorem provides a set of *contact points* in $\partial P \cap \partial E$ whose convex hull already contains the scaled ellipsoid $\frac{1}{\sqrt{d}}E$. Moreover, the number of necessary contact points is at most $\frac{d}{2}(d+3)$, implying the above statement.

So we apply Theorem 4.1 to Q (with center x_0 instead of $\mathbf{0}$) and obtain a list of points $x_1, \dots, x_k \in \mathrm{vert}(C_I)$ with $k \le \frac{1}{2}d(d+3)$ such that

$$C_I \subseteq Q \subseteq \mathrm{conv}(x_0 \pm \lceil\sqrt{d}\rceil \cdot (x_0 - x_j) \mid j \in [k]) =: Q'.$$

Now it is not difficult to cover C_I with parallelepipeds of the form

$$\Pi(J) := \left\{x_0 + \sum_{j \in J} \mu_j (x_j - x_0) \mid |\mu_j| \le \lceil\sqrt{d}\rceil \ \ \forall j \in J\right\}$$

with $J \subseteq [k]$ and $\{x_j - x_0 \mid j \in J\}$ linearly independent. To see this take any point $x \in Q'$. By Carathéodory's Theorem, x lies already in the convex hull of x_0 plus at most d affinely independent vertices of Q', thus there is a subset of indices $J \subseteq [k]$ of size $|J| \leq d$ and signs $\varepsilon_j \in \{\pm 1\}$ with $x \in \text{conv}(\{x_0\} \cup \{x_0 + \varepsilon_j \lceil \sqrt{d} \rceil \cdot (x_j - x_0) \mid j \in J\})$. Then clearly $x \in \Pi(J)$.

Finally it remains to show that all parallelepipeds $\Pi(J)$ are still in P. Let $x = x_0 + \sum_{j \in J} \mu_j (x_j - x_0)$ with $|\mu_j| \leq \lceil \sqrt{d} \rceil$, then for any constraint $i \in [m]$, we have

$$b_i - A_i x \geq \underbrace{b_i - A_i x_0}_{\geq \alpha_{j_i}} - \sum_{j \in J} \underbrace{|\mu_j|}_{\leq \lceil \sqrt{d} \rceil} \cdot \underbrace{|A_i x_j - A_i x_0|}_{\leq \alpha_{j_i+1} - \alpha_{j_i} \leq \frac{\alpha_{j_i}}{d^2}} \geq 0.$$

Finally observe that the number of subsets J of size at most d is $(\frac{1}{2}d(d+3))^d = d^{O(d)}$ which then gives the desired bound.

Now let us argue how to make this constructive in time $N^{O(1)}$. For each cell C, we list the vertices of the integer hull C_I in time $d^{O(d)} m^{O(d)} (\log \Delta)^{O(d)}$ by Theorem 3.2. Computing the minimum volume ellipsoid containing all those vertices can be done using semidefinite programming in time polynomial in the encoding length of the vertices of C_I. The contact points can be inferred from the dual solution of this SDP and the associated parallelepipeds can be easily computed. □

Note that one could have used the following simpler arguments to obtain a weaker, but still polynomial bound: every cell C_I has polynomially many vertices, hence it can be partitioned into polynomially many simplices. Then each simplex can be extended to a parallelepiped, whose union again covers C_I.

As a side remark, the partitioning with shifted hyperplanes was used before e.g. in [CHKM92] to bound the number of extreme points of $\text{conv}(P \cap \mathbb{Z}^d)$.

The next lemma says why parallelepipeds are so useful. Namely the weight of any point in it can be almost completely redistributed to its vertices.

LEMMA 4.2. *Given an integral parallelepiped Π with vertices $X := vert(\Pi)$. Then for any $x^* \in \Pi \cap \mathbb{Z}^d$ and $\lambda^* \in \mathbb{Z}_{\geq 0}$ there is an integral vector $\mu \in \mathbb{Z}_{\geq 0}^{\Pi \cap \mathbb{Z}^d}$ such that*

(1) $\lambda^* x^* = \sum_{x \in \Pi \cap \mathbb{Z}^d} \mu_x x$

(2) $|supp(\mu) \backslash X| \leq 2^d$

(3) $\mu_x \in \{0, 1\}$ $\forall x \notin X$.

Proof. Let $\Pi = \{v_0 + \sum_{i=1}^k \alpha_i v_i \mid |\alpha_i| \leq 1 \ \forall i = 1, \ldots, k\}$ where v_0 is the (not necessarily integral) center of Π.

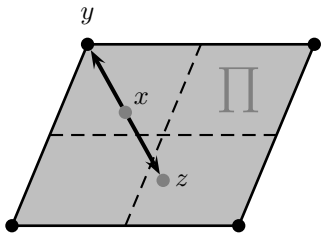


Figure 4: Weight of y is redistributed to vertex in parallelepiped.

Consider a vector μ that satisfies (1) and minimizes the potential function $\sum_{x \notin X} \mu_x$ (i.e. the weight that lies on non-vertices of Π). We claim that μ also satisfies (2) and (3).

First consider the case that there is some point x that is not a vertex and has $\mu_x \geq 2$. We write $x = v_0 + \sum_{i=1}^k \alpha_i v_i$ with $|\alpha_i| \leq 1$. Let[4] $y := v_0 + \sum_{i=1}^k \text{sign}(\alpha_i) \cdot v_i$ be the vertex of Π that we obtain by rounding α_i to ± 1, see Figure 4. Note that the mirrored point $z = x + (x - y) = v_0 + \sum_{i=1}^k (2\alpha_i - \text{sign}(\alpha_i)) \cdot v_i$ lies in Π as well and is also integral. As $x = \frac{1}{2}(y + z)$, we can reduce the weight on x by 2 and add 1 to μ_y and μ_z. We obtain again a vector that satisfies (1), but the weight $\sum_{x \notin X} \mu_x$ has decreased.

So it remains to see what happens when all vectors in $(\Pi \cap \mathbb{Z}^d) \backslash X$ carry weight at most 1. Well, if these are at most 2^d, then we are done. Otherwise, we can reiterate the arguments from Lemma 3.1. There will be 2 points of the same parity, which can be joined to create a new point carrying weight at least 2 and part of this weight can be redistributed to a vertex. This shows the claim. □

Now we simply combine Lemmas 4.1, 3.1 and 4.2.

Proof. [Proof of Structure Theorem 2.3] We choose X as the $N = m^d d^{O(d)} (\log \Delta)^d$ many vertices of parallelepipeds $\mathbf{\Pi}$ that are constructed in Lemma 4.1 in running time $N^{O(1)}$ (there is an extra 2^d factor, that accounts for the maximum number of vertices per parallelepiped; this is absorbed by the O-notation). Now consider any vector $a \in \text{int.cone}(P \cap \mathbb{Z}^d)$. By Lemma 3.1 there is a vector $\mu \in \mathbb{Z}_{\geq 0}^{P \cap \mathbb{Z}^d}$ with $|\text{supp}(\mu)| \leq 2^d$ and $a = \sum_x \mu_x \cdot x$. For every x with $\lambda_x > 0$ we consider a parallelepiped $\Pi \in \mathbf{\Pi}$ with $x \in \Pi \cap \mathbb{Z}^d$. Then we use Lemma 4.2 to redistribute the weight from x to the vertices of Π. For each parallelepiped, there are at most 2^d non-vertices with a weight of 1. In the case

[4] Recall that $\text{sign}(\alpha) = \begin{cases} 1 & \alpha \geq 0 \\ -1 & \alpha < 0 \end{cases}$

in which a vector is used by several parallelepipeds, we can further redistribute its weight to the vertices of one of the involved parallelepipeds. This process terminates as the total weight on X keeps increasing. We denote the new solution by λ. As we are using at most 2^d parallelepipeds, we have $|\text{supp}(\lambda) \cap X| \leq 2^d \cdot 2^d$ and $|\text{supp}(\lambda) \backslash X| \leq 2^d \cdot 2^d$. □

5 Proof of the main theorem

Now that we have the Structure Theorem, the claim of Theorem 2.2 is easy to show.

Proof. [Proof of Main Theorem 2.2] Let $P = \{x \in \mathbb{R}^d \mid Ax \leq b\}$ and $Q = \{x \in \mathbb{R}^d \mid \tilde{A}x \leq \tilde{b}\}$ be the given polytopes. Here we assume that the coefficients in the inequality description are integral and the numbers in A, b and $\tilde{A}, \tilde{b}$ are bounded in absolute value by Δ and $\tilde{\Delta}$, respectively.

We compute the set X of size at most $N := m^d d^{O(d)} (\log \Delta)^d$ from Theorem 2.3 for the polytope P in time $N^{O(1)}$. Now let $y^* \in \text{int.cone}(P \cap \mathbb{Z}^d) \cap Q$ be an unkown target vector. Then we know by Theorem 2.3 that there is a vector $\lambda^* \in \mathbb{Z}_{\geq 0}^{P \cap \mathbb{Z}^d}$ such that $\sum_{x \in P \cap \mathbb{Z}^d} \lambda_x x = y^*$, $|\text{supp}(\lambda^*) \cap X| \leq 2^d$, $|\text{supp}(\lambda^*) \backslash X| \leq 2^d$ and $\lambda^*_x \in \{0, 1\}$ for $x \in (P \cap \mathbb{Z}^d) \backslash X$.

At the expense of a factor $N^{2^{2d}}$ we guess the subset $X' = X \cap \text{supp}(\lambda^*)$[5]. At the expense of another factor $2^{2d} + 1$ we guess the number $k = \sum_{x \notin X'} \lambda^*_x \in \{0, \ldots, 2^{2d}\}$ of extra points. Now we can set up an integer program with few variables. We use variables λ_x for $x \in X'$ to determine the correct multiplicities of the points in X. Moreover, we have variables $x_1, \ldots, x_k \in \mathbb{Z}^d_{\geq 0}$ to determine which extra points to take with unit weight. Additionally we use a variable $y \in \mathbb{Z}^d$ to denote the target vector in polytope Q. The ILP is then of the form

$$\begin{aligned} Ax_i &\leq b \quad \forall i = 1, \ldots, k \\ \sum_{x \in X'} \lambda_x x + \sum_{i=1}^{k} x_i &= y \\ \tilde{A}y &\leq \tilde{b} \\ \lambda_x &\in \mathbb{Z}_{\geq 0} \quad \forall x \in X' \\ x_i &\in \mathbb{Z}^d \quad \forall i = 1, \ldots, k \end{aligned}$$

and given that we made the guessing correctly, this system has a solution. The number of variables is $|X'| + (k+1)d \leq 2^{O(d)}$ and the number of constraints is $km + d + \tilde{m} + |X'|d = 2^{O(d)} m + \tilde{m}$ as well. Note that the largest coefficient is at most $\Delta' := \max\{d! \cdot \Delta^d, \tilde{\Delta}\}$. Hence the system can be solved in time $(2^{O(d)})^{2^{O(d)}} \cdot (2^{O(d)} m + \tilde{m})^{O(1)} \cdot (\log \Delta')^{O(1)}$ via Theorem 3.1. The total running time is hence of the form $\text{enc}(P)^{2^{O(d)}} \cdot \text{enc}(Q)^{O(1)}$. □

Note that it is crucial that the integer combination is taken w.r.t. a set $X = P \cap \mathbb{Z}^d$ that is closed under taking convex combinations. Without this assumption, even for $d = 1$ and $Q = \{a\}$, the test $\text{int.cone}(X) \cap Q \neq \emptyset$ is **NP**-hard as one could define X as the set of numbers in a partition instance.

We can easily generalize this theorem to the case that we can select points from *several* polytopes. In fact, it even works if we can select from sets that are *integer projections* of convex sets, which will turn out to be very useful for our scheduling applications.

COROLLARY 5.1. *Let $P_1, \ldots, P_n$ with $P_i \subseteq \mathbb{R}^{d+d_i}$ be polytopes in inequality form, $c \in \mathbb{Z}^n$ be a cost vector and $Q \subseteq \mathbb{R}^d$ be a target polytope. Define $X_i := \{x \in \mathbb{Z}^d \mid \exists y \in \mathbb{Z}^{d_i} : (x, y) \in P_i\}$. Then the optimization problem*

$$\begin{aligned} \min \quad & \sum_{i \in [n]} c_i \sum_{x \in X_i} \lambda_{i,x} \\ s.t. \quad & \sum_{i \in [n]} \sum_{x \in X_i} \lambda_{i,x} x \in Q \\ & \lambda_{i,x} \in \mathbb{Z}_{\geq 0} \; \forall i \in [n] \; \forall x \in X_i \end{aligned}$$

can be solved in time $(\tilde{m} + \log \Delta)^{2^{O(\tilde{d}+n)}}$ where Δ is the largest coefficient appearing in the input, $\tilde{d} := d + n + \sum_{i=1}^n d_i + 1$ and $\tilde{m}$ is the total number of inequalities describing $P_1, \ldots, P_n$.

Proof. We can assume that $P_i \cap \mathbb{Z}^{d+d_i} \neq \emptyset$, otherwise the polytope can be removed from the list. Moreover, by binary search it suffices to find a solution of cost at most δ, given that there is one. Simply define the polytope $\tilde{P}$ as all vectors $(x, \gamma, x_1, y_1, z_1, \ldots, x_n, y_n, z_n) \in \mathbb{R}^{\tilde{d}}$ satisfying

$$\begin{aligned} (x_i, y_i) &\in P_i \qquad \forall i \in [n] \\ \begin{pmatrix} x \\ \gamma \end{pmatrix} &\{\leq \atop \geq\} \begin{pmatrix} x_i \\ c_i \end{pmatrix} \{+ \atop -\} \Delta(1 - z_i) \cdot \mathbf{1} \quad \forall i \in [n] \\ \sum_{i=1}^{n} z_i &= 1; \quad z \geq \mathbf{0} \end{aligned}$$

Observe that $\tilde{P}$ has dimension $\tilde{d}$ and $O(\tilde{m} + dn)$ constraints. The set of integer vectors (x, γ) that are integer projections of $\tilde{P}$ is exactly $\bigcup_{i=1}^n (X_i \times \{c_i\})$. We apply Theorem 2.2 to $\tilde{P}$ and $\tilde{Q} := Q \times [0, \delta] \subseteq \mathbb{R}^{\tilde{d}-d-1}$ and the solution satisfies the claim. □

[5] Actually we know that X' consists of the vertices of at most 2^d parallelepipeds, thus it suffices to incorporate a factor of N^{2^d}, but the improvement would be absorbed by the O-notation later, anyway.

6 Bin Packing and Cutting Stock

In the *cutting stock* problem, we have again a bin packing instance with sizes $s_1, \dots, s_d \in [0,1]$ and multiplicity a_i of item i. Additionally we have a list of m *bin types*, where bin type $j \in [m]$ has capacity w_j and cost c_j. The study of this problem goes back at least to the 1960's to the classical paper of Gilmore and Gomory [GG61].

COROLLARY 6.1. *The cutting stock problem with d different item types and m different bin types can be solved in time $(\log \Delta)^{2^{O(d+m)}}$ where Δ is the largest number in the input.*

Proof. Simply define $P_j := \{x \in \mathbb{R}^d_{\geq 0} \mid s^T x \leq w_j\}$ for $j \in \{1, \dots, m\}$ and $Q = \{a\}$ and apply Corollary 5.1. □

Recall that a polynomial algorithm was unknown even for $m = 1$ and $d = 3$.

7 Applications to scheduling

In the following we consider a scheduling instance of d different job types and m different machine types. A copy of job $j \in [d]$ has a machine type dependent *release time* of r_{ij} on machine type i, as well as a *deadline* d_{ij} and a *processing time* p_{ij}. We have $a_j \in \mathbb{N}$ many copies of job type j and each machine type i has a cost c_i. Again we assume that all input data is integral. Our goal is to assign all jobs to machines such that all jobs meet their deadline and the cumulated cost of used machines is minimized. We will show that for constant d and m, this assignment problem is solvable in polynomial time. In fact this holds for *non-preemptive* scheduling as well as for scheduling with *preemption* (but without migration).

Note that the case of just $m = 1$ machine type and jobs with identical release times $r_j = 0$ and deadlines $d_j = B$ is equivalent to bin packing (for both, preemptive and non-preemptive scheduling). It seems clear, how to handle the extension of non-trivial release times and deadlines: We simply write down a polytope P such that the vectors $x \in P \cap \mathbb{Z}^d$ define precisely the multi-set of jobs that can be scheduled on a single machine. In fact, it turns out that this is not difficult to do for preemptive scheduling, but more tricky for the setting without preemption.

7.1 Preemptive scheduling. First, let us focus on preemptive scheduling without migration. Note that once the assignment to machines is done, the *Earliest-Deadline First policy* (EDF) gives an optimum schedule [Der74].

Consider a single machine of type i and a vector $x \in \mathbb{Z}^d_{\geq 0}$ of jobs and we wonder how to determine whether the jobs in x can be scheduled on a single machine, i.e. how to test if the EDF schedule of a set of jobs containing x_j copies of job j will meet all the deadlines. If we consider a time interval $[t_1, t_2]$ then it is clear that the total running time of all jobs that have both, release time and deadline in $[t_1, t_2]$ cannot be larger than the length $t_2 - t_1$, otherwise the schedule must be infeasible. In fact, for the EDF-scheduling policy, this is also a sufficient condition[6]. Moreover, it is clear that one does not need to consider *all* time intervals, but just those whose end points lie in the set $T := \{r_{ij}, d_{ij} \mid j \in [d]\}$ of critical points. Thus we can define P_i as the set of vectors $x \in \mathbb{R}^d_{\geq 0}$ such that

$$\sum_{\substack{j \in [d]: \\ r_{ij}, d_{ij} \in [t_1, t_2]}} x_j p_{ij} \leq t_2 - t_1 \quad \forall t_1, t_2 \in T : t_1 \leq t_2. \tag{7.2}$$

Observe that a job vector $x \in \mathbb{Z}^d_{\geq 0}$ can be scheduled on a single machine of type i if and only if $x \in P_i$.

THEOREM 7.1. *Given a vector $a \in \mathbb{Z}^d_{\geq 0}$ of d different job types with release times r_{ij}, deadlines d_{ij} and running times p_{ij} on m different machine types with cost c_i for a machine of type $i \in [m]$. Then one can find an optimum job assignment minimizing the total machine cost under preemptive scheduling in time $(\log \Delta)^{2^{O(d+m)}}$ where Δ is the largest number in the input.*

Proof. We choose polytopes $P_1, \dots, P_m$ as defined in (7.2), each one with d dimensions and described by $O(d^2)$ many constraints, hence we have $O(md^2)$ constraints in total. Then we use Cor. 5.1 to compute the optimum solution in time $(O(md^2) + \log \Delta)^{2^{O(d+m)}} = (\log \Delta)^{2^{O(d+m)}}$. □

Note that the number of different processor schedules returned by the algorithm is bounded by $2^{O(d+m)}$.

7.2 Non-preemptive scheduling. Next, we consider scheduling without preemption. In contrast to the preemptive case, even in the single machine case (i.e. $OPT = m = c_1 = 1$) finding a feasible non-preemptive schedule is **NP**-hard [GJ79] (for general d, but $a_j = 1$). Again, we want to first investigate the case that we have a job vector $x \in \mathbb{Z}^d_{\geq 0}$ and a single machine of some type to schedule all these jobs. For the moment, let us abbreviate the release times, deadlines and processing

[6] This can be easily derived from Hall's condition for the existence of perfect matchings in bipartite graphs and the optimality of EDF.

times with r_j, d_j and p_j. We will see that for fixed d a schedule can be found in polynomial time. For notational convinience we add a dummy job with running time $p_0 = 1$, $r_0 = 0$ and $d_0 := \Delta$ and multiplicity $x_0 := \Delta - \sum_{j=1}^{d} p_j x_j$. Now we can assume that there is no idle time in the schedule.

Let $T := \{r_j, d_j \mid j \in [d]\} = \{t_1, \ldots, t_{2d}\}$ be the $2d$ critical points sorted so that $t_1 \leq \ldots \leq t_{2d}$. The crucial observation is that in a feasible schedule, we can arbitrarily permute jobs that have both start and end time in an interval $[t_k, t_{k+1}]$. Let us imagine that the schedule is *cyclic* in the sense that the schedule processes first some copies of job type 0, then some jobs of type 1, and so on until type d; then the scheduler starts again with jobs of type 0. The interval from a job 0 interval to the beginning of the next job 0 interval is called a *cycle*. Note that the number of copies of job j that are scheduled in a cycle can very well be 0, so indeed such a cyclic schedule trivially exists. Moreover, we want to restrict that a job of type j is only allowed to be scheduled in a cycle if the *complete* cycle is contained in $[r_j, d_j]$. But again this restriction is achievable as we could split cycles if needed.

Now consider the schedule with the least number of cycles. Following our earlier observation it is clear that whenever 2 cycles are completely contained in some interval $[t_k, t_{k+1}]$ of consecutive points, then we could also join them. Thus we can assume that the schedule contains exactly $4d$ many cycles (maybe some have length 0).

We introduce an auxiliary variable y_{jk} which tells us how many copies of job j are processed in the kth cycle. Additionally we have a binary variable z_{jk} telling us whether jobs of type j can be processed in the kth cycle. Moreover, the kth cycle runs in $[\tau_{k-1}, \tau_k]$ (with $\tau_0 := 0$). Then the polytope P whose integral points correspond to feasible schedules can be defined as

(7.3)

$$\begin{array}{rcll}
x_j & = & \sum_{k=1}^{4d} y_{jk} & \forall j \in \{0, \ldots, d\} \\
\tau_k & = & \sum_{\ell \leq k} \sum_{j=0}^{d} p_j y_{j\ell} & \forall k \in [4d] \\
y_{jk} & \leq & \Delta \cdot z_{jk} & \forall j \in [d] \ \forall k \in [4d] \\
\tau_{k-1} & \geq & r_j - \Delta(1 - z_{jk}) & \forall j \in [d] \ \forall k \in [4d] \\
\tau_k & \leq & d_j + \Delta(1 - z_{jk}) & \forall j \in [d] \ \forall k \in [4d] \\
x_0 & = & \Delta - \sum_{j=1}^{d} x_j p_j & \\
y_{jk}, \tau_k & \geq & 0 & \forall j \in \{0, \ldots, d\} \\
 & & & \forall k \in [4d] \\
z_{jk} & \in & [0,1] & \forall j \in [d] \ \forall k \in [4d].
\end{array}$$

A vector $x \in \mathbb{Z}^d_{\geq 0}$ can be non-preemptively scheduled if and only if there are integral x_0, τ, y, z such that $(x, x_0, \tau, y, z) \in P$.

THEOREM 7.2. *Given is a vector $a \in \mathbb{Z}^d_{\geq 0}$ of d different job types with release times r_{ij}, deadlines d_{ij} and running times p_{ij} on m different machine types with cost c_i. Suppose all numbers are integral and bounded by Δ. Then one can compute an optimum non–preemptive job assignment minimizing the total machine cost in time $(\log \Delta)^{2^{O(d^2 m)}}$.*

Proof. We define the polytopes $P_1, \ldots, P_m$ according to (7.3) and apply Cor. 5.1 in order to compute an optimum solution. As each of the m polytopes has $O(d^2)$ many variables and constraints, hence the running time is bounded by $(\log \Delta)^{O(d^2 m)}$. □

One might be tempted to wonder whether the number of variables could be reduced at the expensive of more constraints, which might still improve the running time. But for non-preemptive scheduling we run into the problem that the set of vectors x that can be scheduled on a single machine is not closed under taking convex combinations[7]. In fact, some additional variables are necessary to write those vectors as integer projection of a convex set.

7.3 Minimizing the number of tardy jobs. So far we have considered the case that we have to schedule all jobs and our objective function has been to minimize the number of machines, weighted by cost. Of course, one can also consider the dual setting in which the number of available machines is given and as many jobs as possible should be scheduled in time.

Our input consists again of d job types, where for each job type j, we have a number a_j of copies and a penalty cost c_j. Moreover, we have m machine types with M_i copies of machine type $i \in [m]$. A job of type j has a machine type dependent release time p_{ij}, deadline d_{ij} and running time p_{ij}. The goal is to schedule the jobs on the machines non-preemptively and we have to pay a penalty c_j for each copy of type j that does not finish in time (which for us means it is not scheduled at all).

We saw in (7.3) that we can define a polytope P_i such that the vectors x that are integer projections are exactly those multisets of jobs that can be scheduled on a single machine of type i.

We can slightly change the polytope so that a multiset of jobs represented by $x \in \mathbb{Z}^d_{\geq 0}$ can be scheduled on a single machine of type i if and only if there is some vector $\bar{y} \in \mathbb{Z}^{\bar{d}}$ ($\bar{d} \leq O(d^2)$) with $(x, c^T x, e_i, \bar{y}) \in P_i$ where

[7] A simple example is the following: consider a set of $d = 3$ job types with $\{(r_j, d_j, p_j) \mid j = 1, 2, 3\} = \{(0, 300, 150), (100, 102, 1), (200, 202, 1)\}$. The vectors $x' = (2, 0, 0)$ and $x'' = (0, 2, 2)$ can both be scheduled in a non-preemptive way. But the convex combination $\frac{1}{2}(x' + x'') = (1, 1, 1)$ cannot be scheduled.

$e_i = (0, \ldots, 0, 1, 0, \ldots, 0) \in \mathbb{R}^m$. The polytope P_i has $O(d^2 + m)$ many variables and constraints. We define $X_i = \{(x, c^T x, e_i) \mid \exists \bar{y} \in \mathbb{Z}^{\bar{d}} : (x, c^T x, e_i, \bar{y}) \in P_i\}$ and choose a target polytope

$$Q = [0, a_1] \times \ldots \times [0, a_d] \times [\delta, \infty] \times \{M_1\} \times \ldots \times \{M_m\}.$$

Then we apply Cor. 5.1 and obtain a solution $\lambda_{i,x}$ with $\sum_{i \in [m]} \sum_{x \in X_i} \lambda_{i,x} x \in [0, a]$ using exactly M_i copies of machine i and the penalty of the scheduled jobs is at least δ. If we perform a binary search on δ, we can maximize the penalties and jobs that are scheduled in time (and hence minimize the penalty of those that are not scheduled).

THEOREM 7.3. *Suppose we are given d job types with a_j copies of job i and m machines with M_i copies of machine i. Moreover each job type j has release time, deadline and running time r_{ij}, d_{ij} and p_{ij} on a machine of type $i \in [m]$ and each job has a penalty c_j for each job that does not meet the deadline. Then an optimum job assignment and schedule that minimize the penalty paid for tardy jobs can be found in time* $(\log \Delta)^{2^{O(d^2 m + m^2)}}$.

8 The Eisenbrand-Shmonin Theorem is tight

In this section, we want to describe an example that shows that the Eisenbrand-Shmonin result described in Lemma 3.1 is tight up to a factor of 2. Fix a dimension $d \geq 2$ and let $k := 2^{d-1}$. Let's define a set $X \subset \mathbb{Z}^d$ of k points as

$$\left\{(1 + x_1, \ldots, 1 + x_{d-1}, (4k)^{1 + \sum_{i=1}^{d-1} 2^{i-1} x_i}) \mid x_i \in \{0, 1\}\right\}.$$

For example, for $d = 3$ we obtain

$$\begin{pmatrix} 1 \\ 1 \\ (4k) \end{pmatrix}, \begin{pmatrix} 2 \\ 1 \\ (4k)^2 \end{pmatrix}, \begin{pmatrix} 1 \\ 2 \\ (4k)^3 \end{pmatrix}, \begin{pmatrix} 2 \\ 2 \\ (4k)^4 \end{pmatrix}.$$

We sort $X = \{a_1, \ldots, a_k\}$ according to their length, i.e. $\|a_i\|_\infty = (4k)^i$ and define $P := \text{conv}(X)$.

LEMMA 8.1. *The integer conic combination $y := a_1 + \ldots + a_k$ is unique, thus there are 2^{d-1} points necessary to obtain y as an integer conic combination of points in $P \cap \mathbb{Z}^d$.*

Proof. First, we observe that there is no other integer point in the convex hull P because of the first $d - 1$ coordinates of the a_i's. In other words $P \cap \mathbb{Z}^d = X$. We want to argue that due to the enormous growth of the last coordinate, each a_i has to be used exactly once in order to obtain y.

For this sake, consider any integer conic combination

$$y = \sum_{i=1}^{k} \lambda_i a_i.$$

Note that $y_j = 3 \cdot 2^{d-2}$ for $j \in \{1, \ldots, d-1\}$, thus $\lambda_i \in \{0, \ldots, 3 \cdot 2^{d-2}\}$ for $i = 1, \ldots, k$. We want to argue that $\lambda_1 = \ldots = \lambda_k = 1$ is the only possibility. Suppose this is not the case and let i^* be the largest index with $\lambda_{i^*} \neq 1$. We claim that if $\lambda_{i^*} = 0$, then the combined vector is too short and if $\lambda_{i^*} \geq 2$, then it is too long. Formally, we consider the difference

$$\begin{aligned}
& \|\sum_{i=1}^{k} (\lambda_i - 1) a_i\|_\infty \\
\geq\ & |\lambda_{i^*} - 1| \cdot \|a_{i^*}\|_\infty - \sum_{i=1}^{i^*-1} \underbrace{\lambda_i}_{\leq 2k} \underbrace{\|a_i\|_\infty}_{=(4k)^i} \\
\geq\ & |\lambda_{i^*} - 1| \cdot (4k)^{i^*} - \sum_{i=1}^{i^*-1} 2k(4k)^i \\
\geq\ & (4k)^{i^*} \Big(|\lambda_{i^*} - 1| - \underbrace{\frac{1}{2} \Big(\sum_{j \geq 0} (4k)^{-j} \Big)}_{\leq 3/2} \Big) \\
\geq\ & (|\lambda_{i^*} - 1| - \frac{3}{4}) \cdot (4k)^{i^*}
\end{aligned}$$

using the reverse triangle inequality and the fact that $2k \geq 4$. But if $\lambda_{i^*} \neq 1$, then the length of this vector is strictly larger than 0, hence λ does not correspond to a valid integer conic combination for y. □

Note that $\|a_k\|_\infty = (4k)^k = 2^{\Theta(d 2^d)}$, hence our construction uses numbers that are doubly-exponentional in d. But an alternative argument of [ES06] based on the pigeonhole principle shows that the support of an integer conic combination can also be bounded by $O(d \cdot \log(d \max\{\|x\|_\infty \mid x \in X\}))$. In other words, any construction with minimal conic support of size $\Omega(2^d)$ must contain integer points with coordinates as large as $2^{\Omega(2^d)}$, so the doubly exponentially large numbers are necessary.

References

[CHKM92] W. J. Cook, M. Hartmann, R. Kannan, and C. McDiarmid. On integer points in polyhedra. *Combinatorica*, 12(1):27–37, 1992.

[Der74] M. L. Dertouzos. Control robotics: The procedural control of physical processes. In *IFIP Congress*, pages 807–813, 1974.

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

[FA05] C. Filippi and A. Agnetis. An asymptotically exact algorithm for the high-multiplicity bin packing problem. *Math. Program.*, 104(1):21–37, 2005.

[Fil07] C. Filippi. On the bin packing problem with a fixed number of object weights. *European Journal of Operational Research*, 181(1):117–126, 2007.

[GG61] P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting-stock problem. *Operations Research*, 9:849–859, 1961.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, New York, 1979.

[Har88] M. Hartmann. Cutting planes and the complexity of the integer hull, 1988.

[Joh48] F. John. Extremum problems with inequalities as subsidiary conditions. In *Studies and Essays Presented to R. Courant on his 60th Birthday, January 8, 1948*, pages 187–204. Interscience Publishers, Inc., New York, N. Y., 1948.

[Joh92] D. S. Johnson. The NP-completeness column: An ongoing guide: The tale of the second prover. *Journal of Algorithms*, 13(3):502–524, September 1992.

[JSO10] K. Jansen and R. Solis-Oba. An OPT + 1 algorithm for the cutting stock problem with constant number of object lengths. In Friedrich Eisenbrand and F. Bruce Shepherd, editors, *Proceedings of the 14th International Conference on Integer Programming and Combinatorial Optimization, IPCO 2010*, pages 438–449, 2010.

[Kan87] R. Kannan. Minkowski's convex body theorem and integer programming. *Math. Oper. Res.*, 12(3):415–440, August 1987.

[KK82] N. Karmarkar and R. M. Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *23rd annual symposium on foundations of computer science (Chicago, Ill., 1982)*, pages 312–320. IEEE, New York, 1982.

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

[Mat02] J. Matoušek. *Lectures on discrete geometry*, volume 212 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 2002.

[MSS97] S. T. McCormick, S. R. Smallwood, and F. C. R. Spieksma. Polynomial algorithms for multiprocessor scheduling with a small number of job lengths. In *SODA*, pages 509–517, 1997.

[Rot13] T. Rothvoß. Approximating bin packing within $O(\log OPT * \log\log OPT)$ bins. *CoRR*, abs/1301.4010, 2013.