Bin packing with fixed number of bins revisited

Klaus Jansen[a,1], Stefan Kratsch[b,4], Dániel Marx[c,2], Ildikó Schlotter[d,3,*]

[a] *Institut für Informatik, Christian-Albrechts-Universität Kiel, 24098 Kiel, Germany*
[b] *Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany*
[c] *Computer and Automation Research Institute, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary*
[d] *Budapest University of Technology and Economics, H-1521 Budapest, Hungary*

Abstract

As Bin Packing is NP-hard already for $k = 2$ bins, it is unlikely to be solvable in polynomial time even if the number of bins is a fixed constant. However, if the sizes of the items are polynomially bounded integers, then the problem can be solved in time $n^{O(k)}$ for an input of length n by dynamic programming. We show, by proving the W[1]-hardness of Unary Bin Packing (where the sizes are given in unary encoding), that this running time cannot be improved to $f(k) \cdot n^{O(1)}$ for any function $f(k)$ (under standard complexity assumptions). On the other hand, we provide an algorithm for Bin Packing that obtains in time $2^{O(k \log^2 k)} + O(n)$ a solution with additive error at most 1, i.e., either finds a packing into $k + 1$ bins or decides that k bins do not suffice.

Keywords: Bin Packing, parameterized complexity, additive approximation, W[1]-hardness

1. Introduction

The aim of this paper is to clarify the exact complexity of Bin Packing for a small fixed number of bins. An instance of Bin Packing consists of a set of rational item sizes, the task is to partition the items into a minimum number of bins with capacity 1. Equivalently, we can define the problem such that the sizes are integers and the input contains an integer B, the capacity of the bins.

Complexity investigations usually distinguish two versions of Bin Packing. In the general version, the item sizes are arbitrary integers encoded in binary, thus they can be exponentially large in the size n of the input. In the *unary* version of the problem, the sizes are bounded by a polynomial of the input size; formally, this version requires that the sizes are given in unary encoding.

*Corresponding author. Tel.: +3614632585. Fax: +3614633157.
Email addresses: kj@informatik.uni-kiel.de (Klaus Jansen), kratsch@cs.uu.nl (Stefan Kratsch), dmarx@cs.bme.hu (Dániel Marx), ildi@cs.bme.hu (Ildikó Schlotter)

[1]Supported by DFG Project, Entwicklung und Analyse von effizienten polynomiellen Approximationsschemata für Scheduling- und verwandte Optimierungsprobleme, Ja 612/14-1.
[2]Research supported by the European Research Council (ERC) grant "PARAMTIGHT: Parameterized complexity and the search for tight complexity results," reference 280152.
[3]Supported by the Hungarian National Research Fund (grant OTKA 67651), and by the European Union and the European Social Fund (grant TÁMOP 4.2.1./B-09/1/KMR-2010-0003).
[4]Present address: Utrecht University, 3508 TB Utrecht, The Netherlands.

In the general (not unary) case, a reduction from PARTITION shows that BIN PACKING is NP-hard [8]. Thus it is hard to decide whether a given set of items can be packed into exactly two bins. Apart from NP-hardness, this has a number of other known implications. First of all, unless P = NP, it is impossible to achieve a better polynomial-time approximation ratio than 3/2, matching the best known algorithm [20].

In contrast, however, there are much better approximation results when the optimum number of bins is larger [7, 14]. Fernandez de la Vega and Lueker [7] found an asymptotic polynomial-time approximation scheme (APTAS) for BIN PACKING with ratio $(1+\epsilon)OPT(I)+1$ and running time $O(n)+f(1/\epsilon)$ (if the items are sorted). To bound the function f, one has to consider the integer linear program (ILP) used implicitly in [7]. This ILP has $2^{O(1/\epsilon \log(1/\epsilon))}$ variables and length $2^{O(1/\epsilon \log(1/\epsilon))} \log n$. Using the algorithm by Lenstra [15] or Kannan [13], this ILP can be solved within time $2^{2^{O(1/\epsilon \log(1/\epsilon))}} O(\log n) \leq 2^{2^{O(1/\epsilon \log(1/\epsilon))}} + O(\log^2 n)$. Thus, the algorithm of [7] can be implemented such that the additive term $f(1/\epsilon)$ in the running time is double exponential in $1/\epsilon$. Setting $\epsilon = \frac{1}{OPT(I)+1}$, this algorithm computes a packing into at most $OPT(I)+1$ bins in time $O(n)+2^{2^{OPT(I)\log(OPT(I))}}$.

Using ideas in [5, 12], the algorithm of Fernandez de la Vega and Lueker can be improved to run in time $O(n)+2^{O(1/\epsilon^3 \log(1/\epsilon))}$. Setting again $\epsilon = \frac{1}{OPT(I)+1}$, we obtain an additive 1-approximation that runs in $O(n)+2^{O(OPT(I)^3 \log(OPT(I)))}$ time.

Karmarkar and Karp [14] gave an asymptotic fully polynomial-time approximation scheme (AFPTAS) that packs the items into $(1+\epsilon)OPT(I)+O(1/\epsilon^2)$ bins. The AFPTAS runs in time polynomial in n and $1/\epsilon$, but has a larger additive term $O(1/\epsilon^2)$. Plotkin, Shmoys and Tardos [17] achieved a running time of $O(n\log(1/\epsilon)+\epsilon^{-6})\log^6(1/\epsilon))$ and a smaller additive term $O(1/\epsilon \log(1/\epsilon))$.

BIN PACKING remains NP-hard in the unary case as well [8]. However, for every fixed k, UNARY BIN PACKING with k bins can be solved in polynomial time: a standard dynamic programming approach gives an $n^{O(k)}$ time algorithm. Although the running time of this algorithm is polynomial for every fixed value of k, it is practically useless even for, say, $k = 10$, as an n^{10} time algorithm is usually not considered efficient. Our first result is an algorithm with significantly better running time that approximates the optimum within an additive constant of 1:

Theorem 1. *There is an algorithm for* BIN PACKING *which computes for each instance I of length n a packing into at most $OPT(I)+1$ bins in time*

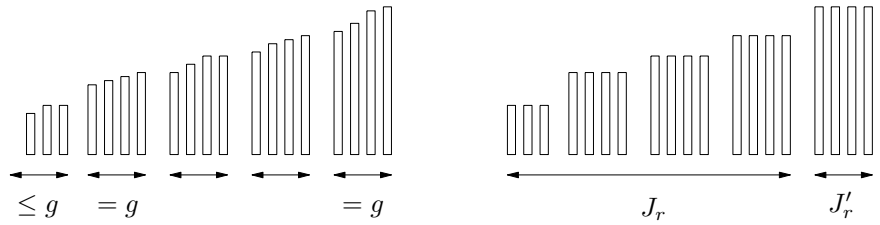
$$2^{O(OPT(I)\log^2 OPT(I))} + O(n).$$

Note that the algorithm works not only for the unary version, but also for the general BIN PACKING as well, where the item sizes can be exponentially large.

It is an obvious question whether the algorithm of Theorem 1 can be improved to an exact algorithm with a similar running time. As the general version of BIN PACKING is NP-hard for $k = 2$, the question makes sense only for the unary version of the problem. By proving that UNARY BIN PACKING is W[1]-hard parameterized by the number k of bins, we show that there is no exact algorithm with running time $f(k) \cdot n^{O(1)}$ for any function $f(k)$ (assuming the standard complexity hypothesis FPT $\neq$ W[1]).

Theorem 2. UNARY BIN PACKING *is* W[1]*-hard, parameterized by the number of bins.*

Thus no significant improvement over the $n^{O(k)}$ dynamic programming algorithm is possible for UNARY BIN PACKING.



(a) The original instance I_r grouped into groups of size $g = \lceil \frac{2^r}{x \log(k)} \rceil$.

(b) The rounded instance J_r and J'_r.

Figure 1: The original and the rounded instances for the interval $(2^{-(r+1)}, 2^{-r}]$.

Furthermore, we also prove that assuming the Exponential Time Hypothesis (saying that 3SAT over n variables can not be solved in $2^{o(n)}$ time) no algorithm can solve Unary Bin Packing in $f(k)n^{o(k/\log k)}$ time for any function f.

2. Additive 1-approximation for Bin Packing in FPT time

This section deals with the following version of Bin Packing: given an integer k and a set I of items with rational item sizes (encoded in binary), and the task is to pack the items into k bins of capacity 1. We prove Theorem 1 by describing an algorithm for this problem which uses at most $k+1$ bins for each I, provided that $OPT(I) = k$, where $OPT(I)$ is the minimum number of bins needed for I.

Our algorithm computes a packing into k or $k+1$ bins. We suppose $k \geq 2$; otherwise we pack all items into a single bin. We divide the instance I into three groups:

$$I_{\text{large}} = \left\{ a \in I \mid \text{size}(a) > \frac{1}{2x} \frac{1}{\log(k)} \right\},$$
$$I_{\text{medium}} = \left\{ a \in I \mid \frac{1}{y} \frac{1}{k} \leq \text{size}(a) \leq \frac{1}{2x} \frac{1}{\log(k)} \right\},$$
$$I_{\text{small}} = \left\{ a \in I \mid \text{size}(a) < \frac{1}{y} \frac{1}{k} \right\},$$

where x, y are positive constants specified later. In the first phase of our algorithm we consider the large items. Since each bin has at most $\lfloor 2x \log(k) \rfloor$ large items and $OPT(I) = k$, the total number of large items in I is at most $k\lfloor 2x \log(k) \rfloor$. Suppose that $I_{\text{large}} = \{a_1, \ldots, a_\ell\}$ where $\ell \leq k\lfloor 2x \log(k) \rfloor$. We can assign large items to bins via a mapping $f : \{1, \ldots, \ell\} \to \{1, \ldots, k\}$. A mapping f is feasible if and only if $\sum_{i|f(i)=j} \text{size}(a_i) \leq 1$ for all $j = 1, \ldots, k$. The total number of feasible mappings or assignments of large items to bins is at most $k^{k\lfloor 2x \log(k) \rfloor} = 2^{O(k \log^2(k))}$, since x is constant. Each feasible mapping f generates a pre-assignment $\text{preass}(b_j) \in [0, 1]$ for the bins $b_j \in \{b_1, \ldots, b_k\}$; i.e. $\text{preass}(b_j) = \sum_{i|f(i)=j} \text{size}(a_i) \leq 1$. Notice that at least one of the $2^{O(k \log^2(k))}$ mappings corresponds to a packing of the large items in an optimum solution.

In the second phase we use a geometric rounding for the medium items. This method was introduced by Karmarkar and Karp [14] for the Bin Packing problem. Let I_r be the set of

all items from I_{medium} whose sizes lie in $(2^{-(r+1)}, 2^{-r}]$. We let $r(0)$ be the integer for which $\frac{1}{2x}\frac{1}{\log(k)} \in (1/2^{r(0)+1}, 1/2^{r(0)}]$. Similarly, we let $r(1)$ be the index with $\frac{1}{yk} \in (2^{-(r(1)+1)}, 2^{-r(1)}]$. Then, the number of indices $r \in \{r(0), \ldots, r(1)\}$ is equal to the number of intervals $(2^{-(r+1)}, 2^{-r}]$ which may contain a medium item. (See Fig. 1(a) for an example I_r where we have divided the set I_r into groups of size $\lceil \frac{2^r}{x\log(k)} \rceil$). Note that the definitions of $r(0)$ and $r(1)$ imply the following bounds:

$$\frac{1}{2^{r(0)}} < \frac{1}{x\log(k)}, \tag{1}$$

$$r(1) \leq \lfloor \log(yk) \rfloor. \tag{2}$$

For each $r \in \{r(0), \ldots, r(1)\}$ let J_r and J'_r be the instances obtained by applying linear grouping with group size $g = \lceil \frac{2^r}{x\log(k)} \rceil$ to I_r. To do this we divide each instance I_r into groups $G_{r,1}, G_{r,2}, \ldots, G_{r,q_r}$ such that $G_{r,1}$ contains the g largest items in I_r, $G_{r,2}$ contains the next g largest items and so on (see Fig. 1(a)). Each group of items is rounded up to the largest size within the group (see also Fig. 1(b)). Let $G'_{r,i}$ be the multi-set of items obtained by rounding the size of each item in $G_{r,i}$. Then, $J_r = \bigcup_{i\geq 2} G'_{r,i}$ and $J'_r = G'_{r,1}$.

Furthermore, let $J = \bigcup J_r$ and $J' = \bigcup J'_r$. Then, $J_r \leq I_r \leq J_r \cup J'_r$ where $\leq$ is the partial order on Bin Packing instances with the interpretation that $I_A \leq I_B$ if there exists a one-to-one function $h : I_A \to I_B$ such that $\text{size}(a) \leq \text{size}(h(a))$ for all items $a \in I_A$. Furthermore, J'_r consists of one group of items with the largest medium items in $(2^{-(r+1)}, 2^{-r}]$. The cardinality of each group (with exception of maybe the smallest group in I_r) is equal to $\lceil \frac{2^r}{x\log(k)} \rceil$.

Lemma 1. *For $k \geq 2$ and $x \geq 4$, we have* $\text{size}(J') \leq \frac{\log(yk)}{x\log(k)}$.

Proof. Each non-empty set J'_r contains at most $\lceil \frac{2^r}{x\log(k)} \rceil$ items each of size at most $1/2^r$. Hence

$$\text{size}(J'_r) \leq \left(\frac{2^r}{x\log(k)} + 1 \right) \frac{1}{2^r} = \frac{1}{x\log(k)} + \frac{1}{2^r}.$$

This implies that the total size of J' is

$$\text{size}(J') = \sum_{r=r(0)}^{r(1)} \text{size}(J'_r) \leq \sum_{r=r(0)}^{r(1)} \left(\frac{1}{x\log(k)} + \frac{1}{2^r} \right).$$

The bound (1) for $k \geq 2$ yields $2^{r(0)} > x\log(k) \geq x$, or equivalently, $r(0) > \log(x\log(k)) \geq \log(x)$. For $x \geq 4$ we obtain $r(0) \geq 3$. Using also the upper bound (2) on $r(1)$, we have that

$$r(1) - r(0) + 1 \leq r(1) - 2 \leq \lfloor \log(yk) \rfloor - 2. \tag{3}$$

Now, by (1) and $\sum_{r=r(0)}^{r(1)} 1/2^r \leq 1/2^{r(0)-1}$, we get

$$\begin{aligned} \text{size}(J') &\leq (\lfloor \log(yk) \rfloor - 2)\frac{1}{x\log(k)} + \sum_{r=r(0)}^{r(1)} \frac{1}{2^r} \\ &\leq \frac{\log(yk) - 2}{x\log(k)} + \frac{1}{2^{r(0)-1}} \leq \frac{\log(yk) - 2}{x\log(k)} + \frac{2}{x\log(k)} \leq \frac{\log(yk)}{x\log(k)}. \end{aligned}$$

This completes the proof. □

The lemma above implies $OPT(J') = 1$ for $x \geq 4$, $k \geq 2$ and $\log(y) \leq (x-1)$, since these items have total size at most 1. A possible choice is $x = 4$ and $y \leq 8$.

By $\bigcup_{r=r(0)}^{r(1)} J_r \leq I_{\text{medium}} \leq \bigcup_{r=r(0)}^{r(1)} (J_r \cup J'_r)$ and $J' = \bigcup_{r=r(0)}^{r(1)} J'_r$ with $OPT(J') = 1$, we obtain:

Lemma 2.

$$OPT(I_{\text{large}} \cup \bigcup_{r=r(0)}^{r(1)} J_r \cup I_{\text{small}}) \leq OPT(I_{\text{large}} \cup I_{\text{medium}} \cup I_{\text{small}})$$

$$OPT(I_{\text{large}} \cup I_{\text{medium}} \cup I_{\text{small}}) \leq OPT(I_{\text{large}} \cup \bigcup_{r=r(0)}^{r(1)} J_r \cup I_{\text{small}}) + 1.$$

Lemma 3. *There are $O(k \log(k))$ different rounded sizes for medium items for $x \geq 4$ and $k \geq 2$.*

Proof. Let $n(I_r)$ be the number of medium items in I_r, and let $m(I_r)$ be the number of groups (or rounded sizes) generated by the linear grouping for I_r. Then,

$$\text{size}(I_r) \geq \frac{1}{2^{r+1}} n(I_r) \geq \frac{1}{2^{r+1}} \left((m(I_r) - 1) \left\lceil \frac{2^r}{x \log(k)} \right\rceil \right).$$

Notice that one group may have less than $\lceil \frac{2^r}{x \log(k)} \rceil$ items. This implies that

$$m(I_r) - 1 \leq \frac{2^{r+1} \text{size}(I_r)}{\left\lceil \frac{2^r}{x \log(k)} \right\rceil}.$$

Using $\lceil a \rceil \geq a$ for $a \geq 0$, we get

$$m(I_r) \leq 2x \log(k) \text{size}(I_r) + 1.$$

For $x \geq 4$ and $k \geq 2$, the bound (3) implies $r(1) - r(0) + 1 \leq \log(yk)$, which gives us that the total number of rounded medium sizes is

$$\sum_{r=r(0)}^{r(1)} m(I_r) \leq \sum_{r=r(0)}^{r(1)} (2x \log(k) \text{size}(I_r) + 1) \leq 2x \log(k) \sum_{r=r(0)}^{r(1)} \text{size}(I_r) + \log(yk).$$

Since all medium items fit into k bins and x, y are constants, $\text{size}(I_{\text{medium}}) = \sum_{r=r(0)}^{r(1)} \text{size}(I_r) \leq k$ and

$$\sum_{r=r(0)}^{r(1)} m(I_r) \leq 2xk \log(k) + \log(yk) = O(k \log(k)).$$

This completes the proof. □

Now we describe the third phase of our algorithm. The rounded medium item sizes lie in the interval $[\frac{1}{yk}, \frac{1}{2x \log(k)}]$ and there are $R = O(k \log(k))$ many different rounded item sizes. For each $j = 1, \ldots, R$ let k_j be the number of items for each rounded item size x_j. Since $x_j \geq \frac{1}{yk}$

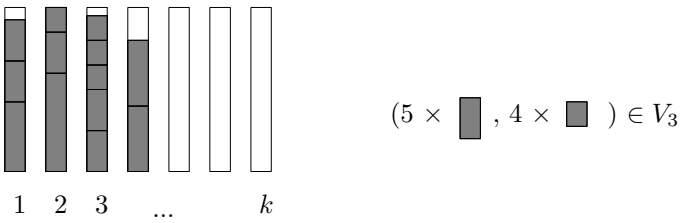


Figure 2: The dynamic program for rounded medium items $J = \cup_r J_r$

and $OPT(I) = k$, the number $k_j \leq k/x_j \leq k^2 y$ for each item size x_j. To describe a packing for one bin b we use a mapping $p : \{1, \ldots, R\} \to \{0, \ldots, yk\}$ where $p(j)$ gives the number of items of size x_j in b. A mapping p is feasible if and only if $\sum_j p(j)x_j + \text{preass}(b) \leq 1$, where preass$(b)$ is the total size of large items assigned to b in the first phase of the algorithm. The total number of feasible mappings for one bin is at most $(yk+1)^{O(k \log(k))} = 2^{O(k \log^2(k))}$. Using a dynamic program we go over the bins from b_1 up to b_k. For each $A = 1, \ldots, k$, we compute a set V_A of vectors $(a_1, \ldots, a_R)$ where a_j gives the number of items of size x_j used for the bins $b_1, \ldots, b_A$ (see also Fig. 2). The cardinality of each set V_A is at most $(k^2 y + 1)^{O(k \log(k))} = 2^{O(k \log^2(k))}$. The update step from one bin to the next (computing the next set V_{A+1}) can be implemented in time

$$2^{O(k \log^2(k))} \cdot 2^{O(k \log^2(k))} \cdot \text{poly}(k) \leq 2^{O(k \log^2(k))}.$$

If there is a solution for our Bin Packing instance I into k bins, then the set V_k contains the vector $(n_1, \ldots, n_R)$ that corresponds to the number of rounded medium item sizes in $\bigcup_{r=r(0)}^{r(1)} J_r$. Notice that the other set $\bigcup_{r=r(0)}^{r(1)} J'_r$ will be placed into the additional bin b_{k+1}. We can also compute a packing of the medium items into the bins as follows. First, we compute all vector sets V_A for $A = 1, \ldots, k$. If for two vectors $a = (a_1, \ldots, a_k) \in V_A$ and $a' = (a'_1, \ldots, a'_k) \in V_{A+1}$ the medium items given by the difference $a' - a$ and the preassigned large items fit into bin b_{A+1}, we store the corresponding pair (a, a') in a set S_{A+1}. By using a directed acyclic graph $D = (V, E)$ with vertex set $V = \{[a, A] | a \in V_A, A = 1, \ldots, k\}$ and $E = \{([a, A], [a', A+1]) | (a, a') \in S_{A+1}, A = 1, \ldots, k-1\}$, we may compute a feasible packing of large and medium rounded items into the bins $b_1, \ldots, b_k$. This can be done via depth first search starting with the vector $(n_1, \ldots, n_R) \in V_k$ that corresponds to the number of rounded medium item sizes. The algorithm to compute the directed acyclic graph and the backtracking algorithm can be implemented in time

$$2^{O(k \log^2(k))}.$$

In the last phase of our algorithm we add the small items via a greedy algorithm to the bins. Consider a process which starts with a given packing of the original large and medium items into the bins $b_1, \ldots, b_{k+1}$. We insert the small items of size at most $\frac{1}{yk}$ with the greedy algorithm Next Fit (NF) into the bins $b_1, \ldots, b_{k+1}$. The correctness of this step is proved in the next lemma. Notice that NF can be implemented in linear time with $O(n)$ operations.

Lemma 4. *If $OPT(I) = k$, $k \geq 2$, and $y \geq 2$, then our algorithm packs all items into at most $k+1$ bins.*

PROOF. Assume by contradiction that we use more than $k+1$ bins for the small items. In this case, the total size of the items packed into the bins is more than $(k+1)(1-\frac{1}{yk}) = k+1-\frac{k+1}{yk}$. Note that $\frac{k+1}{yk} \leq \frac{k+1}{2k} < 1$ by $y \geq 2$ and $k \geq 2$. Thus, the total size of the items is larger than k, yielding $OPT(I) > k$. □

The algorithm for BIN PACKING for $OPT(I) = k$ works as follows:

(1) Set $x = 4$, $y = 2$ and divide the instance I into three groups I_{large}, I_{medium}, and I_{small}.

(2) Assign the large items to k bins considering all different feasible pre-assignments.

(3) Use geometric rounding on the sizes of the medium items; for each interval $(2^{-(r+1)}, 2^{-r}]$ apply linear grouping with group size $\lceil \frac{2^r}{x \log(k)} \rceil$ to the item set I_r and compute rounded item sets J_r and J'_r.

(4) For each pre-assignment apply the dynamic program to assign the medium items in $\bigcup J_r$ to the bins $b_1, \ldots, b_k$, and place the set $\bigcup_j J'_r$ into the additional bin b_{k+1}.

(5) Take a feasible packing into $k+1$ bins for one pre-assignment (there is one by $OPT(I) = k$), replace the rounded items by their original sizes and afterwards assign the small items via a greedy algorithm to the bins $b_1, \ldots, b_{k+1}$.

3. Parameterized hardness of Bin Packing

The aim of this section is to prove that UNARY BIN PACKING is W[1]-hard, parameterized by the number k of bins. In this version of BIN PACKING, we are given a set of integer item sizes encoded in unary, and two integers b and k. The task is to decide whether the items can be packed into k bins of capacity b.

To prove the W[1]-hardness of this problem when parameterized by the number of bins, we use the hardness of an intermediate problem, a variant of UNARY BIN PACKING involving vectors of constant length c and bins of varying sizes.

Let $[c] = \{1, \ldots, c\}$ for any $c \in \mathbb{N}$, and let $\mathbb{N}^c$ be the set of vectors with c coordinates, each in $\mathbb{N}$. We use boldface letters for vectors. Given vectors $\mathbf{v}, \mathbf{w} \in \mathbb{N}^c$, we write $\mathbf{v} \leq \mathbf{w}$, if $v^j \leq w^j$ for each $j \in [c]$, where v^j is the j-th coordinate of $\mathbf{v}$. As usual, the addition of vectors is defined as the addition of their corresponding components.

For a fixed c, we will call the following problem c-UNARY VECTOR BIN PACKING. We are given a set of n items having sizes $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_n$ with each $\mathbf{s}_i \in \mathbb{N}^c$ encoded in unary, and k vectors $\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_k$ from $\mathbb{N}^c$ representing bin sizes. The task is to decide whether $[n]$ can be partitioned into k sets $J_1, J_2, \ldots, J_k$ such that $\sum_{h \in J_i} \mathbf{s}_h \leq \mathbf{B}_i$ for each $i \in [k]$.

Before proving Lemma 6 saying that c-UNARY VECTOR BIN PACKING for any constant c can be reduced to UNARY BIN PACKING by a parameterized reduction, we need some simple observations about vectors. For integers b and c, we let the *representation with base b* of a vector $\mathbf{v} \in \mathbb{N}^c$ be $R(\mathbf{v}, b) = v^1 + v^2 b + \cdots + v^c b^{c-1}$.

Proposition 5. *(1) Suppose that for given vectors $\mathbf{v}, \mathbf{w} \in \mathbb{N}^c$ and for any $j \in [c-1]$ it holds that $v^j < b$ and $w^j < b$; the last coordinates of the vectors can be unbounded. Then $R(\mathbf{v}, b) = R(\mathbf{w}, b)$ holds if and only if $\mathbf{v} = \mathbf{w}$.*
(2) For any $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k \in \mathbb{N}^c$ it holds that $\sum_{i=1}^k R(\mathbf{v}_i, b) = R(\sum_{i=1}^k \mathbf{v}_i, b)$.

PROOF. We prove (1) by induction on c, the case $c = 1$ is trivial. So suppose $c > 1$ and $R(\mathbf{v}, b) = R(\mathbf{w}, b)$. First, observe $v^1 = R(\mathbf{v}, b) \bmod b = R(\mathbf{w}, b) \bmod b = w^1$. By $v^1 < b$ and $w^1 < b$, this implies $v^1 = w^1$.

Now, let $\bar{\mathbf{v}}$ (and $\bar{\mathbf{w}}$) be the vector in $\mathbb{N}^{c-1}$ whose coordinates are the last $c-1$ coordinates of $\mathbf{v}$ ($\mathbf{v}$, respectively). Clearly, we have $R(\mathbf{v}, b) = v^1 + bR(\bar{\mathbf{v}}, b)$ and $R(\mathbf{v}, b) = v^1 + bR(\bar{\mathbf{v}}, b)$. Using also $v^1 = w^1$, we obtain $R(\bar{\mathbf{v}}, b) = R(\bar{\mathbf{w}}, b)$. By induction, this yields $\bar{\mathbf{v}} = \bar{\mathbf{w}}$, which finishes the proof of (1).

Claim (2) can be shown by an easy calculation: $\sum_{i\in[k]} R(\mathbf{v}_i, b) = \sum_{i\in[k]} \sum_{j\in[c]} v_i^j b^{j-1} = \sum_{j\in[c]} b^{j-1}(\sum_{i\in[k]} v_i^j) = R(\sum_{i\in[k]} \mathbf{v}_i, b)$. □

Lemma 6 below shows that Theorem 2 follows from the W[1]-hardness of c-UNARY VECTOR BIN PACKING, for any fixed c. In Section 3.1, we introduce two concepts, k-non-averaging and k-sumfree sets, that will be useful tools in the hardness proof. The reduction itself appears in Section 3.2.

Lemma 6. *For every fixed integer $c \geq 1$, there is a parameterized reduction from c-UNARY VECTOR BIN PACKING to UNARY BIN PACKING, where both problems are parameterized by the number of bins.*

PROOF. Let $\mathcal{I}$ be an instance of c-UNARY VECTOR BIN PACKING containing n items with item sizes $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_n$ in $\mathbb{N}^c$, and k vectors $\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_k$ from $\mathbb{N}^c$ representing bin sizes. Clearly, we can assume that $\mathbf{B}_i \leq \sum_{h\in[n]} \mathbf{s}_h$ for each $i \in [k]$, as otherwise the instance $\mathcal{I}$ is trivially solvable.

We construct an instance $\mathcal{I}'$ of UNARY BIN PACKING as follows. Let N be the maximum value of the integers $\{s_i^j \mid i \in [n], j \in [c]\}$, and let $b = nN + 1$. For each item appearing in $\mathcal{I}$ with size $\mathbf{s}_i$ for some $i \in [n]$, we create a corresponding item in $\mathcal{I}'$ whose size is the representation of $\mathbf{s}_i$ with base b. Let also B_i be the representation of $\mathbf{B}_i$ with base b, let B be the maximum among the numbers B_i, $i \in [k]$, and let us introduce a set T of k new items having sizes $T_i = 2B + 1 - B_i$ for each $i \in [k]$. We set the bin size to $2B + 1$. Note that the number of items in the constructed instance $\mathcal{I}'$ is $n + k$.

We claim that the UNARY BIN PACKING instance $\mathcal{I}'$ has a solution with k bins if and only if $\mathcal{I}$ is feasible. It is straightforward to see that any solution of $\mathcal{I}$ can be transformed into a solution of $\mathcal{I}'$ by putting the item of size T_i into the i-th bin, as well as those items of $\mathcal{I}'$ with sizes $R(\mathbf{s}_h, b)$ for which the original item of size $\mathbf{s}_h$ is put in the i-th bin. For the other direction, note that each item in T has size at least $B + 1$, because $B_i \leq B$ for all $i \in [k]$ by the definition of B. Hence, two items from T cannot be put in the same bin, so it can be assumed without loss of generality that a solution of $\mathcal{I}'$ puts the item having size T_i into the i-th bin. Therefore, the total size of the items not in T contained in the i-th bin is at most $2B + 1 - T_i = B_i$. Using Proposition 5 and the fact $b > nN$, it follows that for any $S \subseteq [n]$ we have $\sum_{h\in S} R(\mathbf{s}_h, b) \leq B_i$ if and only if $\sum_{h\in S} \mathbf{s}_h \leq \mathbf{B}_i$. Hence, the items of $\mathcal{I}$ corresponding to the items of $\mathcal{I}'$ in the i-th bin have total size at most $\mathbf{B}_i$.

Finally, observe that the maximal integer in this instance is smaller than $2B + 1 \leq 2(b^c) + 1 = 2(nN + 1)^c + 1$. Clearly, $N < |\mathcal{I}|$ (as $\mathcal{I}$ is encoded in unary), and c is a fixed constant, so this is polynomial in $|\mathcal{I}|$. Thus, the whole construction takes polynomial time. □

3.1. Non-averaging and sumfree sets

We call a set A k-non-averaging, if for any subset of at most k elements in A it holds that the arithmetic mean of these elements is not contained in A. Such sets have already been studied by several researchers [6, 2, 1, 4]. Below we describe a method that constructs a k-non-averaging set

having n elements. Although, up to our knowledge, the construction presented does not appear in the literature in this form, it applies only standard techniques.

We will also use the concept of k-sumfree sets. A set B is called k-sumfree, if no two different subsets of B having cardinality $k' \leq k$ can have the same sum. Such sets have been studied extensively in the literature, also under the name B_k-sequences (which stands for integer sequences whose elements form a k-sumfree set) [9]. For more on this area, see [11, 10, 18, 19].

Non-averaging sets. Given an integer k, we are going to construct a set $\mathcal{A}$ containing n non-negative integers with the following property: for any k elements $a_1, a_2, \ldots, a_k$ in $\mathcal{A}$ it holds that their arithmetical mean $\frac{1}{k}\sum_{i=1}^{k} a_i$ can only be contained in $\mathcal{A}$ if all of them are equal, i.e. $a_1 = a_2 \cdots = a_k$. Sets having this property will be called *k-non-averaging*.

First, let us fix an arbitrary integer d. (In fact, it will suffice to assume $d = 2$, but for completeness, we present the case for an arbitrary d.) Depending on d, let us choose m to be the smallest integer for which $m^d \geq n$, i.e. let $m = \lceil n^{1/d} \rceil$. We construct a set $\mathcal{X}$ containing each vector $(x_1, x_2, \ldots, x_d, y)$ where $0 \leq x_i \leq m-1$ for all $i \in [d]$ and $\sum_{i=1}^{d} x_i^2 = y$. Clearly, $|\mathcal{X}| = m^d$, so in particular, $|\mathcal{X}| \geq n$.

Lemma 7. *If $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k$ and $\mathbf{v}$ are elements of $\mathcal{X}$ and $\mathbf{v} = \frac{1}{k}\sum_{i=1}^{k} \mathbf{u}_i$, then $\mathbf{u}_1 = \mathbf{u}_2 = \cdots = \mathbf{u}_k = \mathbf{v}$.*

Proof. Let us define $\mathbf{t}_i$ for each $i \in [k]$ such that $\mathbf{u}_i = \mathbf{v} + \mathbf{t}_i$. Note that $\mathbf{t}_i$ is a vector whose coordinates are possibly negative. We will show that all of its coordinates must be equal to zero.

Using that the j-th coordinates of $k\mathbf{v}$ and $\sum_{i=1}^{k} \mathbf{u}_i$ are equal for each $j = 1, \ldots, d, d+1$, we obtain the followings.

First, if $j \in [d]$, then $\sum_{i=1}^{k}(v^j + t_i^j) = kv^j$, resulting in $\sum_{i=1}^{k} t_i^j = 0$. Second, observe that the last coordinate of $\mathbf{u}_i$ can be written in the form $\sum_{j=1}^{d}(v^j + t_i^j)^2$. Thus, we obtain $k\sum_{j=1}^{d}(v^j)^2 = \sum_{i=1}^{k}\sum_{j=1}^{d}(v^j + t_i^j)^2$, yielding the equation $\sum_{i=1}^{k}\sum_{j=1}^{d}((t_i^j)^2 + 2v^j t_i^j) = 0$. Using $\sum_{i=1}^{k} t_i^j = 0$ we have $\sum_{i=1}^{k}\sum_{j=1}^{d} 2v^j t_i^j = 0$, implying $\sum_{i=1}^{k}\sum_{j=1}^{d}(t_i^j)^2 = 0$. But this can only hold if $t_i^j = 0$ for every $i \in [k]$ and $j \in [d]$. This altogether means that $v^j = u_i^j$ for each $j \in [d]$, so by $\mathbf{v}, \mathbf{u}_i \in \mathcal{X}$ we have $\mathbf{v} = \mathbf{u}_i$ for each i, proving the lemma. □

Next, we construct a non-averaging set $\mathcal{A}$ from the set $\mathcal{X}$.

Lemma 8. *If $b = k(m-1)+1$, then the set $\mathcal{A} = \{v^1 + v^2 b + \cdots + v^d b^{d-1} \mid \mathbf{v} \in \mathcal{X}\}$ is k-non-averaging. Moreover, the largest element N in $\mathcal{A}$ is smaller than $4d(2k)^d n^{1+2/d}$, and $\mathcal{A}$ can be constructed in time linear in $O(2^d nN)$.*

Proof. Note that for any vector $\mathbf{v} = (x_1, x_2, \ldots, x_d, y) \in \mathcal{X}$ we have $x_i \leq m-1$ if $i \in [d]$. This shows that the sum of any k vectors in $\mathcal{X}$ must have coordinates at most $k(m-1)$, except maybe for the last coordinate. Therefore, if we set b such that $b > k(m-1)$, then by combining claims (1) and (2) of Prop. 5, we obtain that for any $k+1$ vectors $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k, \mathbf{v}$ in $\mathcal{X}$ we have that $\sum_{i=1}^{k} \mathbf{u}_i = k\mathbf{v}$ holds if and only if $\sum_{i=1}^{k} R(\mathbf{u}_i, b) = kR(\mathbf{v}, b)$. Thus, the bound on b guarantees that the set $\mathcal{A} = \{R(\mathbf{v}, b) \mid \mathbf{v} \in \mathcal{X}\}$ is indeed k-non-averaging by Lemma 7.

Let us upper bound the maximal integer in $\mathcal{A}$ as a function of d, k, and n. First, it should be clear that the maximal element N of $\mathcal{A}$ is $R(\mathbf{w}, b)$ for the vector $\mathbf{w} = (m-1, \ldots, m-1, d(m-1)^2)$.

Thus, $N = (m-1)\frac{b^d-1}{b-1} + d(m-1)^2 b^d < db^d m^2 < d(km)^d m^2$ by our assumption on b. Using the definition of $m = \lceil n^{1/d} \rceil$, we have $m \leq n^{1/d} + 1 \leq 2n^{1/d}$. Thus, we obtain $N < 4d(2k)^d n^{1+2/d}$.

Finally, it is clear that any element of $\mathcal{X}$ can be constructed in $O(d \log^2 m)$ time, and its representation can be obtained in $O(b^d d^2 \log m) = O(N)$ time. By $|\mathcal{X}| = m^d \leq 2^d n$, we get that $\mathcal{X}$ and $\mathcal{A}$ can be constructed in $O(2^d nN)$ time. □

Using Lemma 8 and setting $d = 2$, we can find a k-non-averaging set $\mathcal{A}$ having n elements such that the maximal element in $\mathcal{A}$ is smaller than $2^5 k^2 n^2 = O(k^2 n^2)$. Also, $\mathcal{A}$ can be constructed in $O(k^2 n^3)$ time.

Sumfree sets. A set $\mathcal{F}$ is *k-sumfree*, if for any two subsets $S_1, S_2 \subseteq \mathcal{F}$ of the same size $k' \leq k$, $\sum_{x \in S_1} x = \sum_{x \in S_2} x$ holds if and only if $S_1 = S_2$.

It is easy to verify that the set $S = \{(k+1)^i \mid 0 \leq i < n\}$ is a k-sumfree set of size n. To see this, assume for the sake of contradiction that there exist two different subsets $S_1, S_2 \subseteq S$ of the same size $k' \leq k$ with $\sum_{x \in S_1} x = \sum_{x \in S_2} x$. Notice that the elements of S are the $(k+1)$-base representations of those 0-1 vectors V_S in $\mathbb{N}^n$ that have exactly one coordinate of value 1. Let us consider the two different subsets $V_1, V_2 \subseteq V_S$ of size k' containing precisely those vectors in V_S whose $(k+1)$-base representations are contained in the sets S_1 and S_2, respectively. Using $\sum_{x \in S_1} x = \sum_{x \in S_2} x$ and claim (2) of Prop. 5, we get $R(\sum_{\mathbf{v} \in V_1} \mathbf{v}, k+1) = R(\sum_{\mathbf{v} \in V_2} \mathbf{v}, k+1)$. As both V_1 and V_2 contain at most k 0-1 vectors and our base is $k+1 > k$, claim (1) of Prop. 5 implies $\sum_{\mathbf{v} \in V_1} \mathbf{v} = \sum_{\mathbf{v} \in V_2} \mathbf{v}$. But since no vector in $\mathbb{N}^n$ can be obtained in more than one different way as the sum of at most k vectors from V_S, this contradicts $V_1 \neq V_2$. Hence, S is indeed k-sumfree. The maximal element in the set S is of course $(k+1)^{n-1}$.

Although this will be sufficient for our purposes, we mention that a construction due to Bose and Chowla [3] shows that a k-sumfree set of size n with maximum element at most $(2n)^k$ can also be found (see also [11], Chapter II). If k is relatively small compared to n, the bound $(2n)^k$ is a considerable reduction on the bound $(k+1)^{n-1}$ of the construction above.

Lemma 9 ([3]). *For any integers n and k, there exists a k-sumfree set having n elements, with the maximum element being at most $(2n)^k$.*

3.2. Hardness of the vector problem

The following lemma contains the main part of the hardness proof. By Lemma 6, it immediately implies Theorem 2.

Lemma 10. 10-Unary Vector Bin Packing *is* W[1]*-hard.*

Proof. We give an FPT reduction from Subgraph Isomorphism. Given graphs G and H, the Subgraph Isomorphism problem asks whether G contains a subgraph which is isomorphic to H. This problem is W[1]-hard when parameterized by the number $|E(H)|$ of edges of H.

Let $G = (V, E)$ and $H = (U, F)$ be an instance of Subgraph Isomorphism. We assume $V = [n]$, $|E| = m$, $U = \{u_1, u_2, \ldots, u_\ell\}$, and $|F| = k$. Thus, the parameter value of the given instance is k. W.l.o.g. we suppose $\ell = |U| \leq k$, as otherwise we can obtain an equivalent problem instance by adding a clique on λ new vertices both to G and to H; setting $\lambda = \max(\ell, 5)$ ensures the smaller graph to have at least as many edges as vertices. We are going to construct an instance $\mathcal{I}$ of 10-Unary Vector Bin Packing with $k + \ell + 1$ bins.

Notation and other ingredients. We will write e_i for the i-th edge of G according to some arbitrary fixed ordering. Let $\mathcal{P}$ denote the set $\{(i,j) \mid u_iu_j \in F \text{ and } i < j\}$, and let $\mathcal{P}'$ denote the set $\{(i,j) \mid u_iu_j \in F \text{ or } i = j\}$. Note that $|\mathcal{P}| = k$ but $|\mathcal{P}'| = 2k + \ell$. For each vertex u_i in H we define $d_i^< = |\{j \mid u_iu_j \in F, j < i\}|$ to denote the number of edges going from u_i to a vertex with smaller index in H, and we also define $d_i^> = |\{j \mid u_iu_j \in F, j > i\}|$.

To define the item sizes, we need to construct an ℓ-non-averaging set $\mathcal{A}$ of size n, using Lemma 8. Let $\mathcal{A}$ contain the elements $a_1 \le a_2 \le \cdots \le a_n$. Taking $d = 2$, by Lemma 8, we know $a_n = O(\ell^2 n^2)$. Let $A = \sum_{v\in[n]} a_v$, clearly $A = O(\ell^2 n^3)$.

We also construct an ℓ-sumfree set $\mathcal{B}$ containing $2k + \ell$ elements, using Lemma 9. Let us index the elements of $\mathcal{B}$ by the pairs contained in $\mathcal{P}'$, so let $\mathcal{B} = \{b_{i,j} \mid (i,j) \in \mathcal{P}'\}$. We also assume that $b_{i_1,j_1} < b_{i_2,j_2}$ holds if and only if (i_1, j_1) is lexicographically smaller than (i_2, j_2). By Lemma 9, we know that $b_{k,k} = O((2k + \ell)^\ell) = O((3k)^\ell)$ holds. Again, we let $B = \sum_{b\in\mathcal{B}} b$, so $B = O((3k)^{\ell+1})$. Moreover, we set $B_i^< = \sum_{(j,i)\in\mathcal{P}} b_{i,j}$ and $B_i^> = \sum_{(i,j)\in\mathcal{P}} b_{i,j}$ for each $i \in [\ell]$.

Item sizes. We are going to define two sets of item sizes in $\mathcal{I}$, contained in the sets S and T. The item sizes in S are further divided into sets $S_{i,j}$ where (i,j) ranges over all pairs in $\mathcal{P}$, i.e. $S = \bigcup_{(i,j)\in\mathcal{P}} S_{i,j}$. Similarly, the item sizes in the set T are further divided into sets $T_{i,j}$ where (i,j) ranges over all pairs in $\mathcal{P}'$. Furthermore, in connection to the sets $T_{i,j}$ we distinguish between three cases, depending on whether $(i,j) \in \mathcal{P}$, or $(j,i) \in \mathcal{P}$, or $1 \le i = j \le \ell$ holds; we define three sets $T^<$, $T^>$, and $T^=$ corresponding to these cases as shown below. Now, the set T is defined by $T = T^= \cup T^< \cup T^>$.

$$\begin{aligned} T^= &= \textstyle\bigcup_{i\in[\ell]} T_{i,i}, \\ T^< &= \textstyle\bigcup_{(i,j)\in\mathcal{P}} T_{i,j}, \text{ and} \\ T^> &= \textstyle\bigcup_{(i,j)\in\mathcal{P}} T_{j,i}. \end{aligned}$$

For each pair $(i,j) \in \mathcal{P}$, $|S_{i,j}| = m$ will hold. Also, for each pair $(i,j) \in \mathcal{P}'$, $|T_{i,j}| = n$ will hold.

Given a pair $(i,j) \in \mathcal{P}$, we put an item $\mathbf{s}_{i,j}(e)$ into $S_{i,j}$ for each edge e in G, so we let $S_{i,j} = \bigcup_{e\in E} \mathbf{s}_{i,j}(e)$. Similarly, given a pair $(i,j) \in \mathcal{P}'$ we put an item $\mathbf{t}_{i,j}(v)$ into $T_{i,j}$ for each vertex v in G, so we let $T_{i,j} = \bigcup_{v\in V} \mathbf{t}_{i,j}(v)$. The exact values of $\mathbf{s}_{i,j}(e)$ and $\mathbf{t}_{i,j}(v)$ are the following 10-vectors:

$$\begin{aligned} \mathbf{s}_{i,j}(e) &= (ki+j, 1, \ 0\ , 0, \ 0\ , 0, \ 0\ , 0, \quad a_x \quad , \quad a_y \quad) && \text{if } (i,j) \in \mathcal{P},\ e = xy \in E, \\ \mathbf{t}_{i,i}(v) &= (\quad 0 \quad , 0, b_{i,i}, 1, \ 0\ , 0, \ 0\ , 0, d^>(i)a_v, d^<(i)a_v) && \text{if } i \in [\ell],\ v \in V, \\ \mathbf{t}_{i,j}(v) &= (\quad 0 \quad , 0, \ 0\ , 0, b_{i,j}, 1, \ 0\ , 0, \quad a_v \quad , \quad 0 \quad) && \text{if } (i,j) \in \mathcal{P},\ v \in V, \\ \mathbf{t}_{j,i}(v) &= (\quad 0 \quad , 0, \ 0\ , 0, \ 0\ , 0, b_{j,i}, 1, \quad 0 \quad , \quad a_v \quad) && \text{if } (i,j) \in \mathcal{P},\ v \in V. \end{aligned}$$

Bin capacities. We define $k + \ell + 1 \le 2k + 1$ bins as follows: we introduce bins $\mathbf{p}_{i,j}$ for each $(i,j) \in \mathcal{P}$, bins $\mathbf{q}_i$ for each $i \in [\ell]$, and one additional bin $\mathbf{r}$. The capacities of the bins $\mathbf{p}_{i,j}$ and $\mathbf{q}_i$ are given below (depending on i and j).

$$\begin{aligned} \mathbf{p}_{i,j} &= (ki+j, 1, \quad 0 \quad , \quad 0 \quad , (n-1)b_{i,j}, n-1, (n-1)b_{j,i}, n-1, \quad A \quad , \quad A \quad) \\ \mathbf{q}_i \ &= (\quad 0 \quad , 0, (n-1)b_{i,i}, n-1, \quad B_i^> \quad , d^>(i), \quad B_i^< \quad , d^<(i), d^>(i)A, d^<(i)A) \end{aligned}$$

Finally, we set the capacity of $\mathbf{r}$ in a way that the total capacity of the bins equals the total size of the items. Hence, any solution must completely fill all bins.

It is easy to see that each component of $\mathbf{r}$ is non-negative. Observe that $|S \cup T| = mk + n(2k + \ell)$, the unary encoding of the item sizes in S needs a total of $O(mk^2\ell + mkA)$ bits, and the unary encoding of the item sizes in T needs a total of at most $O(nB + kA)$ bits. By the bounds on A and B, the reduction given is indeed an FPT reduction.

Main idea. At a high-level abstraction, we think of the constructed instance as follows. First, a bin $\mathbf{q}_i$ requires $n-1$ items from $T_{i,i}$, which means that we need all items from $T_{i,i}$, except for one item $\mathbf{t}_{i,i}(w)$. Choosing such an index $w \in [n]$ for each $i \in [\ell]$ will correspond to choosing ℓ vertices from G. Next, we have to fill up the bin $\mathbf{q}_i$, by taking altogether $d^<(i) + d^>(i)$ items from $T^<$ and $T^>$ in a way such that the sum of their last two coordinates equals the last two coordinates of $\mathbf{t}_{i,i}(w)$. The sumfreeness of $\mathcal{B}$ and the non-averaging property of $\mathcal{A}$ will imply that each of the chosen items must be of the form $\mathbf{t}_{i,j}(w)$ or $\mathbf{t}_{j,i}(w)$ for some j.

This can be thought of as "copying" the information about the chosen vertices, since as a result, each bin $\mathbf{p}_{i,j}$ will miss only those items from $T_{i,j}$ and from $T_{j,i}$ that correspond to the i-th and j-th chosen vertex in G. Suppose $\mathbf{p}_{i,j}$ contains all items from $T_{i,j}$ and $T_{j,i}$ except for the items, say, $\mathbf{t}_{i,j}(h_a)$ and $\mathbf{t}_{j,i}(h_b)$. Then, we must fill up the last two coordinates of $\mathbf{p}_{i,j}$ exactly, by choosing one item from $S_{i,j}$. But choosing the item $\mathbf{s}_{i,j}(e)$ for some $u_iu_j \in F$ will only do if the edge corresponding to $e \in E$ connects the vertices h_a and h_b in G corresponding to vertices u_i and u_j in H. This ensures that whenever u_i and u_j are connected in H, the i-th and the j-th chosen vertices in G are also connected. This means that H is indeed a subgraph of G.

Correctness. Now, let us show formally that $\mathcal{I}$ is solvable if and only if G contains a subgraph isomorphic to H. Clearly, $\mathcal{I}$ is solvable if and only if each of the bins can be filled exactly. Thus, a solution for $\mathcal{I}$ means that the items in $S \cup T$ can be partitioned into sets $\{P_{i,j} \mid (i,j) \in \mathcal{P}\}$, $\{Q_i \mid i \in [\ell]\}$, and R such that

$$\mathbf{p}_{i,j} = \sum_{\mathbf{v} \in P_{i,j}} \mathbf{v} \quad \text{for each } (i,j) \in \mathcal{P} \tag{4}$$

$$\mathbf{q}_i = \sum_{\mathbf{v} \in Q_i} \mathbf{v} \quad \text{for each } i \in [\ell], \text{ and} \tag{5}$$

$$\mathbf{r} = \sum_{\mathbf{v} \in R} \mathbf{v}. \tag{6}$$

Next, we are going to count the number of items from S, $T^=$, $T^<$, and $T^>$ that are contained in one of the sets among $\{P_{i,j} \mid (i,j) \in \mathcal{P}\}$, $\{Q_i \mid i \in [\ell]\}$, or R in an arbitrary solution. To argue in general, let X denote the set of items that are contained in some particular bin $\mathbf{x}$. Observing the second, fourth, sixth, and eighth coordinates of the items in $S \cup T$ and the capacity of the bin $\mathbf{x}$, we can immediately count the number of items from S, $T^=$, $T^<$, and $T^>$ that are contained in X. For example, the only items having a non-zero element on the second coordinate are the items in S; since each of them have value 1 on this coordinate, we obtain that each bin $\mathbf{p}_{i,j}$ contains exactly one element from S (as the second coordinate of $\mathbf{p}_{i,j}$ is 1), and each bin $\mathbf{q}_i$ contains zero items from S (as the second coordinate of $\mathbf{q}_i$ is 0). Table 1 shows the information obtained by this argument for each possible bin $\mathbf{x}$.

Direction $\Rightarrow$. First, we argue that if G contains H as a subgraph, then $\mathcal{I}$ is solvable. Suppose that H appears as a subgraph of G on the vertices $c_1, c_2, \ldots, c_\ell$ in V such that for each $i \in [\ell]$ the vertex u_i can be mapped to c_i, meaning that $u_iu_j \in F$ implies $c_ic_j \in E$. Let $e_{i,j}$ be the edge c_ic_j of G for each $(i,j) \in \mathcal{P}$. We set $P_{i,j}$ for each $(i,j) \in \mathcal{P}$ and Q_i for each $i \in [\ell]$ as follows, letting R include all the remaining items.

$$P_{i,j} = \{\mathbf{t}_{i,j}(v) \mid v \neq c_i\} \cup \{\mathbf{t}_{j,i}(v) \mid v \neq c_j\} \cup \{\mathbf{s}_{i,j}(e_{i,j})\}. \tag{7}$$

$$Q_i = \{\mathbf{t}_{i,j}(c_i) \mid (i,j) \in \mathcal{P}\} \cup \{\mathbf{t}_{j,i}(c_i) \mid (i,j) \in \mathcal{P}\} \cup \{\mathbf{t}_{i,i}(v) \mid v \neq c_i\}. \tag{8}$$

	S	$T^=$	$T^<$	$T^>$
$P_{i,j}$ for some $(i,j) \in \mathcal{P}$	1	0	$n-1$	$n-1$
Q_i for some $i \in [\ell]$	0	$n-1$	$d^>(i)$	$d^<(i)$
R	$(m-1)k$	ℓ	0	0

Table 1: The number of items contained in the different item sets S, $T^=$, $T^<$, and $T^>$, depicted for each bin. In the table, each row corresponds to the items of one bin. Some entry for a particular bin $\mathbf{x}$ shows how many items packed by some arbitrary solution into $\mathbf{x}$ are contained in the item set corresponding to the given column.

It is easy to see that the sets $P_{i,j}$ for some $(i,j) \in \mathcal{P}$ and the sets Q_i for some $i \in [\ell]$ are all pairwise disjoint. Thus, in order to verify that this indeed yields a solution, it suffices to check that (4) and (5) hold, since in that case (6) follows from the way $\mathbf{r}$ is defined. For any $(i,j) \in \mathcal{P}$, using

$$\begin{aligned}
\sum_{v \neq c_i} \mathbf{t}_{i,j}(v) &= \sum_{v \neq c_i} (\ 0\ ,0,0,0,\ b_{i,j}\ ,\ 1\ ,\ 0\ ,\ 0\ ,\ a_v\ ,\ 0\ ) \\
&= (\ 0\ ,0,0,0,(n-1)b_{i,j},n-1,\ 0\ ,\ 0\ ,A-a_{c_i},\ 0\ ), \\
\sum_{v \neq c_j} \mathbf{t}_{j,i}(v) &= \sum_{v \neq c_j} (\ 0\ ,0,0,0,\ 0\ ,\ 0\ ,\ b_{j,i}\ ,\ 1\ ,\ 0\ ,\ a_v\ ) \\
&= (\ 0\ ,0,0,0,\ 0\ ,\ 0\ ,(n-1)b_{j,i},n-1,\ 0\ ,A-a_{c_j}), \\
\mathbf{s}_{i,j}(e_{i,j}) &= (ki+j,1,0,0,\ 0\ ,\ 0\ ,\ 0\ ,\ 0\ ,\ a_{c_i}\ ,\ a_{c_j}\ ),
\end{aligned}$$

we get (4) by the definition of $P_{i,j}$. To see (5), we only have to use the definition of Q_i, and sum up the equations below:

$$\begin{aligned}
\sum_{j:(i,j)\in\mathcal{P}} \mathbf{t}_{i,j}(c_i) &= \sum_{j:(i,j)\in\mathcal{P}} (0,0,\ 0\ ,\ 0\ ,b_{i,j},\ 1\ ,0\ ,\ 0\ ,\ a_{c_i}\ ,\ 0\ ) \\
&= (0,0,\ 0\ ,\ 0\ ,B_i^>,d^>(i),\ 0\ ,\ 0\ ,\ d^>(i)a_{c_i}\ ,\ 0\ ), \\
\sum_{j:(i,j)\in\mathcal{P}} \mathbf{t}_{j,i}(c_i) &= \sum_{j:(i,j)\in\mathcal{P}} (0,0,\ 0\ ,\ 0\ ,\ 0\ ,\ 0\ ,\ b_{j,i},\ 1\ ,\ 0\ ,\ a_{c_i}\ ) \\
&= (0,0,\ 0\ ,\ 0\ ,\ 0\ ,\ 0\ ,B_i^<,d^<(i),\ 0\ ,\ d^<(i)a_{c_i}\ ), \\
\sum_{v \neq c_i} \mathbf{t}_{i,i}(v) &= \sum_{v \neq c_i} (0,0,\ b_{i,i}\ ,\ 1\ ,\ 0\ ,\ 0\ ,\ 0\ ,\ 0\ ,\ d^>(i)a_v\ ,\ d^<(i)a_v\ ) \\
&= (0,0,(n-1)b_{i,i},n-1,\ 0\ ,\ 0\ ,\ 0\ ,\ 0\ ,d^>(i)(A-a_{c_i}),d^<(i)(A-a_{c_i})).
\end{aligned}$$

Direction $\Leftarrow$. To prove the other direction, suppose that a solution exists, meaning that some sets $\{P_{i,j} \mid (i,j) \in \binom{[k]}{2}\}$, $\{Q_i \mid i \in [k]\}$ and R fulfill the conditions of (4), (5), and (6). We show that this implies that G contains a subgraph isomorphic to H.

Let us observe that $r^3 = \sum_{i\in[\ell]} b_{i,i}$. This means that R contains exactly ℓ vectors from $\bigcup_{i\in[\ell]} T_{i,i}$ such that the third coordinate of their sum is $\sum_{i\in[\ell]} b_{i,i}$. But since $\mathcal{B}$ is ℓ-sumfree, this can only happen if R contains exactly one vector from each of $T_{1,1}, T_{2,2}, \ldots, T_{\ell,\ell}$. Let these vectors be $\{\mathbf{t}_{i,i}(c_i) \mid i \in [\ell]\}$. We will argue that the vertices $\{c_i \mid i \in [\ell]\}$ prove that H is a subgraph of G, by showing $c_ic_j \in E$ for each $(i,j) \in \mathcal{P}$.

Using $q_i^3 = (n-1)b_{i,i}$, Table 1, and $b_{1,1} < b_{2,2} < \cdots < b_{\ell,\ell}$ we obtain that Q_i must contain every item in $T_{i,i} \setminus \{\mathbf{t}_{i,i}(c_i)\}$, for each $i \in [\ell]$. Also, we know that Q_i must contain $d^>(i)$ items from $T^<$ and $d^<(i)$ items from $T^>$, so from the values of q_i^5 and q_i^7 and the fact that $\mathcal{B}$ is ℓ-sumfree (note $d^<(i) + d^>(i) < \ell$), we also obtain that Q_i must contain exactly one item from each of the sets $T_{i,j}$ where $(i,j) \in \mathcal{P}$ or $(j,i) \in \mathcal{P}$. Observe that apart from these $(n-1) + d^<(i) + d^>(i)$ items, Q_i cannot contain any other items.

Now, fix some i and note that the last two coordinates of the sum $\sum_{v \neq c_i} \mathbf{t}_{i,i}(v)$ are exactly $d^>(i)(A - a_{c_i})$ and $d^<(i)(A - a_{c_i})$. Since the last two coordinates of $\mathbf{q}_i$ are $d^>(i)A$ and $d^<(i)A$, we get that $\sum_{\mathbf{v} \in Q_i \setminus T_{i,i}} \mathbf{v}$ must have $d^>(i)a_{c_i}$ and $d^<(i)a_{c_i}$ at the last two coordinates. As argued above, $Q_i \setminus T_{i,i}$ contains exactly one item from each of the sets $T_{i,j}$ where $(i,j) \in \mathcal{P}$ or $(j,i) \in \mathcal{P}$. Let us define h_j for each $j \neq i$ where $(i,j) \in \mathcal{P}'$ such that $T_{i,j} \cap Q_i = \{t_{i,j}(h_j)\}$. Then we obtain $\sum_{(i,j) \in \mathcal{P}} a_{h_j} = d^>(i)a_{c_i}$ and $\sum_{(j,i) \in \mathcal{P}} a_{h_j} = d^<(i)a_{c_i}$. But as $\mathcal{A}$ is ℓ-non-averaging, this yields $h_j = c_i$ for each j. This means that (8) holds.

Next, let us consider the set $P_{i,j}$ for some $(i,j) \in \mathcal{P}$. First, the first two coodinates of $\mathbf{p}_{i,j}$ imply that $P_{i,j}$ must contain exactly one element of $S_{i,j}$. Let us define $e_{i,j}$ such that $P_{i,j} \cap S_{i,j} = \mathbf{s}_{i,j}(e_{i,j})$. Furthermore, Table 1 shows that $P_{i,j}$ must contain $(n-1)$ items from both of the sets $T^<$ and $T^>$. Recall that $\{\mathbf{t}_{i,j}(c_i) \mid (i,j) \in \mathcal{P}', i \neq j\} \subseteq \bigcup_{i \in [\ell]} Q_i$ by the result (8), so from each set $T_{i,j}$ where $i \neq j$ there can be at most $n-1$ items contained in $P_{i,j}$. Using $p_{i,j}^5 = (n-1)b_{i,j}$ and $p_{i,j}^7 = (n-1)b_{j,i}$, and taking into account the ordering of the elements of $\mathcal{B}$, it follows that (7) holds as well.

Finally, let us focus on the last two coordinates of the sum $\sum_{\mathbf{v} \in P_{i,j}} \mathbf{v}$ for some $(i,j) \in \mathcal{P}$. The sum of the vectors in $T_{i,j} \setminus \{\mathbf{t}_{i,j}(c_i)\}$ has $A - a_{c_i}$ and 0 as the last two coordinates, and similarly, the sum of the vectors in $T_{j,i} \setminus \{\mathbf{t}_{j,i}(c_j)\}$ has 0 and $A - a_{c_j}$ in the last two coordinates. From this, (7) and the definition of $\mathbf{p}_{i,j}$ yield that $\mathbf{s}_{i,j}(e_{i,j})$ must contain a_{c_i} and a_{c_j} in the last two coordinates. But by the definition of $S_{i,j}$, this can only hold if (c_i, c_j) is an edge in G. This proves the second direction of the correctness of the reduction. □

Lemmas 6 and 10 together prove Theorem 2, establishing the W[1]-hardness of Unary Bin Packing when parameterized by the number of bins. Hence, unless the standard complexity-theoretic assumption W[1] $\neq$ FPT fails, there is no algorithm for Unary Bin Packing with running time of the form $f(k)n^{O(1)}$ for some funcion f.

Next, in Theorem 3 we show an even stronger lower bound on the running time of any algorithm solving Unary Bin Packing, assuming the Exponential Time Hypothesis (ETH). This hypothesis says that there is no $2^{o(n)}$ time algorithm for n-variable 3SAT. Note that since ETH is a stronger assumption than W[1] $\neq$ FPT, Theorem 3 is not a direct consequence of Theorem 2. Instead, the proof of Theorem 3 relies on a result by Marx [16] and makes use of the fact that the parameterized reductions presented in Lemmas 6 and 10 are from Subgraph Isomorphism.

Theorem 3. *There is no algorithm solving the* Unary Bin Packing *problem in $f(k)n^{o(k/\log k)}$ time for some function f, where k is the number of bins in the input and n is the input length, unless ETH fails.*

Proof. Suppose that we are given an input I_1 of the Subgraph Isomorphism problem where n_1 is the length of the input given and k_1 is the number of edges of the smaller graph. By the reductions present in Lemmas 6 and 10, we can construct an equivalent instance I_2 of Unary Bin Packing in $f(k_1)n_1^{O(1)}$ time with $k_2 = \Theta(k_1)$ bins. This shows that if Unary Bin Packing can be solved in $f'(k_2)|I_2|^{o(k_2/\log k_2)}$ time for some function f', then Subgraph Isomorphism can be

solved in $f''(k_1)n_1^{o(k_1/\log k_1)}$ time for some function f''. By Theorem 1.5 in [16], this would imply that ETH fails. □

We would like to remark that both Theorems 2 and 3 remain true for the special version of Unary Bin Packing where the total size of the items equals the total capacity of the bins. Finally, let us mention that, by the generality of Unary Bin Packing, our hardness results might be useful when proving hardness of other problems as well.

4. Conclusion

We studied the parameterized complexity of Bin Packing where the parameter is the number k of bins. We presented an algorithm with running time $2^{O(k\log^2 k)} + O(n)$ that provides a packing with additive error 1, where n denotes the input size. This result is an improvement over the additive 1-approximation that can be derived using the APTAS given by Fernandez de la Vega and Lueker [7]. Improving the running time further, perhaps to $2^{O(k)}n^{O(1)}$ time, is an obvious direction for further research. Let us recall here that it is a long-standing open question whether there is a polynomial-time additive 1-approximation for Bin Packing.

We also examined Unary Bin Packing where each item size is encoded in unary. We focused on the question whether the well-known dynamic programming algorithm running in $n^{O(k)}$ time can be improved considerably. We proved the W[1]-hardness of this problem when parameterized by k. This rules out the possibility of giving an $f(k)n^{O(1)}$ algorithm for any computable function f, supposing W[1]≠FPT. The reduction at some point uses certain number-theoretic constructions (k-non-averaging and k-sumfree sets), which might be useful in other hardness proofs. Using the Exponential Time Hypothesis (which is a stronger assumption than W[1]≠FPT), we proved that there is no algorithm for Unary Bin Packing running in time $n^{o(k/\log k)}$. One can notice that our results leave open the question whether there exists an algorithm for Unary Bin Packing that runs in $n^{o(k)}$ time. Also, it might be possible to prove stronger parameterized hardness results for this problem such as W[2]-hardness, or even W[t]-hardness for every t.

Acknowledgement

We are grateful to Imre Ruzsa for explaining us the techniques used for the construction of k-non-averaging sets.

[1] H. L. Abbott. Extremal problems on non-averaging and non-dividing sets. *Pacific J. Math.*, 91:1–12, 1980.

[2] N. Alon and I. Z. Ruzsa. Non-averaging subsets and non-vanishing transversals. *J. Comb. Theory, Ser. A*, 86(1):1–13, 1999.

[3] R. C. Bose and S. Chowla. Theorems in the additive theory of numbers. *Comment. Math. Helv.*, 37(1):141–147, 1962-63.

[4] A. P. Bosznay. On the lower estimation of non-averaging sets. *Acta Math. Hung.*, 53:155–157, 1989.

[5] F. Eisenbrand and G. Shmonin. Caratheodory bounds for integer cones. *OR Letters*, 34:564–568, 2006.

[6] P. Erdős and E. G. Straus. Non-averaging sets II. *Combinatorial Theory and its Applications, Vol. II, Colloquia Mathematica Societatis János Bolyai*, 4:405–411, 1970.

[7] W. Fernandez de la Vega and G. S. Lueker. Bin packing can be solved in within $1+\epsilon$ in linear time. *Combinatorica*, 1:349–355, 1981.

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

[9] S. W. Graham. B_h sequences. In *Analytic number theory, Vol. 1 (1995)*, volume 138 of *Progr. Math.*, pages 431–449. Birkhäuser Boston, Boston, MA, 1996.

[10] R. K. Guy. *Unsolved problems in number theory*. Problem Books in Mathematics. Springer-Verlag, New York, third edition, 2004.

[11] H. Halberstam and K. F. Roth. *Sequences*. Springer-Verlag, New York, 1983.

[12] K. Jansen. An EPTAS for scheduling jobs on uniform processors: using an MILP relaxation with a constant number of integral variables. In *ICALP 09: 36th International Colloquium on Automata, Languages and Programming*, pages 562–573, 2009.

[13] R. Kannan. Minkowski's convex body theorem and integer programming. *Math. of OR*, 12:415–440, 1987.

[14] N. Karmarkar and R. M. Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *FOCS 1982: 23rd IEEE Symposium on Foundations of Computer Science*, pages 312–320, 1982.

[15] H. W. Lenstra. Integer programming with a fixed number of variables. *Math. of OR*, 8:538–548, 1983.

[16] D. Marx. Can you beat treewidth? *Theory of Computing*, 6(1):85–112, 2010.

[17] S. A. Plotkin, D. B. Shmoys, and E. Tardos. Fast approximation algorithms for fractional packing and covering problems. *Math. of OR*, 20:257–301, 1995.

[18] I. Z. Ruzsa. Solving a linear equation in a set of integers. I. *Acta Arith.*, 65(3):259–282, 1993.

[19] I. Z. Ruzsa. Solving a linear equation in a set of integers. II. *Acta Arith.*, 72(4):385–397, 1995.

[20] D. Simchi-Levi. New worst-case results for the bin-packing problem. *Naval Res. Logist.*, 41(4):579–585, 1994.