# Bin packing with fixed number of bins revisited

Klaus Jansen[1]*, Stefan Kratsch[2], Dániel Marx[3]**, and Ildikó Schlotter[4]***

[1] Institut für Informatik, Christian-Albrechts-Universität Kiel, 24098 Kiel, Germany
[2] Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany
[3] Tel Aviv University, Israel
[4] 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.

## 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, and 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.

In the general (not unary) case, a reduction from PARTITION shows that BIN PACKING is NP-hard [7]. 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 [16].

In contrast, however, there are much better approximation results when the optimum number of bins is larger [4, 12]. De la Vega and Lueker [4] found an

* Research supported by the EU project AEOLUS, Contract Number 015964.
** Supported by ERC Advanced Grant DMMCA.
*** Supported by the Hungarian National Research Fund (OTKA 67651).

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 [4]. 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 [13] or Kannan [11], 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 [4] 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 [6, 10], the algorithm of 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 [12] 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 [14] 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 [7]. However, for every fixed $k$, Unary Bin Packing 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. From the perspective of parameterized complexity, the general (not unary) BIN PACKING is not even known to be contained in the class XP, when parameterized by the number of bins.

Finally, let us mention that the existence of a polynomial-time algorithm with additive error of 1 (or any other constant) is a fundamental open problem, discussed for example in [5, Section 2.2.9]. Such a polynomial-time algorithm would be a significant breakthrough even in the unary case. Our algorithm shows that obtaining such an approximation is easy in the unary case for fixed number of bins. Thus an important consequence of our result is that any hardness proof ruling out the possibility of constant additive error approximation for the unary version has to consider instances with an unbounded number of bins.

For proofs omitted due to lack of space, marked by a $\star$, see the full paper.

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

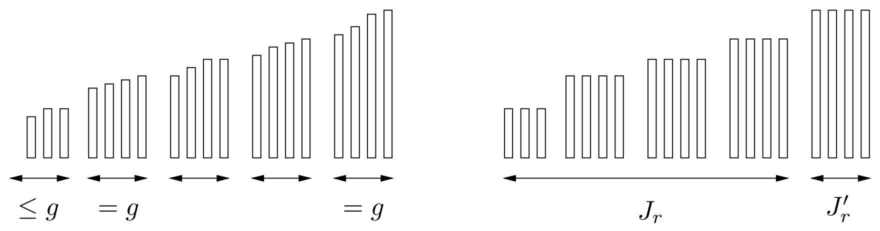
This section deals with the following version of BIN PACKING: given an integer $K$ and a set $I$ of items with rational item sizes (encoded in binary), and the task is to pack the items into $K$ bins of capacity 1. We prove Theorem 1 by describing an algorithm for this problem which uses at most $K+1$ 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:

$$\begin{aligned} I_{large} &= \{a \in I \mid size(a) > \tfrac{1}{2x}\tfrac{1}{\log(K)}\}, \\ I_{medium} &= \{a \in I \mid \tfrac{1}{y}\tfrac{1}{K} \leq size(a) \leq \tfrac{1}{2x}\tfrac{1}{\log(K)}\}, \\ I_{small} &= \{a \in I \mid size(a) < \tfrac{1}{y}\tfrac{1}{K}\}, \end{aligned}$$

where $x, y$ are 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 $I_{large} = \{a_1, \ldots, a_\ell\}$ 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 $\sum_{i|f(i)=j} size(a_i) \leq 1$ for all $j = 1, \ldots, K$. The total number of feasible mappings or assignments of large items to bins is at most $K^{K\lfloor 2x\log(K)\rfloor} = 2^{O(K\log^2(K))}$. Each feasible mapping $f$ generates a pre-assignment $preass(b_j) \in [0, 1]$ for the bins $b_j \in \{b_1, \ldots, b_K\}$; i.e. $preass(b_j) = \sum_{i|f(i)=j} size(a_i) \leq 1$. 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 [12] for the BIN PACKING problem. Let $I_r$ be the set of all items from $I_{medium}$ whose sizes lie in $(2^{-(r+1)}, 2^{-r}]$ where $2^r > x\log(K)$ or equivalently $\frac{1}{x}\frac{1}{\log(K)} > \frac{1}{2^r}$ (see Fig. 1(a) for an example $I_r$ where we have divided the set $I_r$ into groups of size $\lceil \frac{2^r}{x\log(K)}\rceil$). Let $r(0)$ be the smallest integer $r$ such that $2^r > x\log(K)$. Then, $2^{r(0)-1} \leq x\log(K)$ and $2^{r(0)} > x\log(K)$. This implies that the interval with the smallest index $r(0)$ contains items of size in $(1/2^{r(0)+1}, 1/2^{r(0)}]$ and that $\frac{1}{2x}\frac{1}{\log(K)} \in (1/2^{r(0)+1}, 1/2^{r(0)}]$.



(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 $J'_r$.

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

For each $r \geq r(0)$ 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 $size(x) \leq size(h(x))$ for all items $x \in I_A$. 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 $size(J') \leq \frac{\log(yK)}{x \log(K)}$.*

*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 $size(J'_r) \leq (\frac{2^r}{x \log(K)} + 1)\frac{1}{2^r} = \frac{1}{x \log(K)} + \frac{1}{2^r}$. This implies that the total size $size(J') = \sum_{r \geq r(0)} size(J'_r) \leq \sum_{r \geq r(0)} (\frac{1}{x \log(K)} + \frac{1}{2^r})$. Let $r(1)$ be the index with $\frac{1}{yK} \in (2^{-(r(1)+1)}, 2^{-r(1)}]$. This implies that $r(1) \leq \lfloor \log(yK) \rfloor$. 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. Since $\frac{1}{x \log(K)} > 1/2^{r(0)}$ and $K \geq 2$, we have $2^{r(0)} > x \log(K) \geq x$ or equivalently $r(0) > \log(x \log(K)) \geq \log(x)$. For $x \geq 4$ we obtain $r(0) \geq 3$.

Thus, the number of such intervals is $r(1) - r(0) + 1 \leq r(1) - 2 \leq \lfloor \log(yK) \rfloor - 2$. Using $\frac{1}{2^{r(0)}} < \frac{1}{x \log(K)}$ and $\sum_{r \geq r(0)} 1/2^r \leq 1/2^{r(0)-1}$, we get

$$\begin{aligned} size(J') &\leq (\lfloor \log(yK) \rfloor - 2)\frac{1}{x \log(K)} + \textstyle\sum_{r \geq r(0)} \frac{1}{2^r} \\ &\leq \frac{\log(yK) - 2}{x \log(K)} + \frac{1}{2^{r(0)-1}} \leq \frac{\log(yK) - 2}{x \log(K)} + \frac{2}{x \log(K)} \leq \frac{\log(yK)}{x \log(K)}. \end{aligned}$$ □

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 $y \leq 8$.

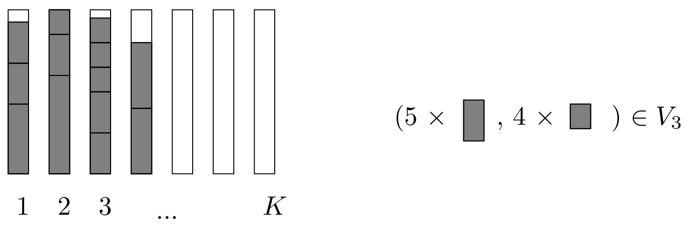


**Fig. 2.** The dynamic program for rounded medium items $J = \cup_j J_r$

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

**Lemma 2.**

$$OPT(I_{large} \cup \bigcup_{r \geq r(0)} J_r \cup I_{small}) \leq OPT(I_{large} \cup I_{medium} \cup I_{small})$$

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

**Lemma 3.** *There are at most $O(K \log(K))$ different rounded sizes for medium items for $x \geq 1$ 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, $size(I_r) \geq \frac{1}{2^{r+1}} n(I_r) \geq \frac{1}{2^{r+1}} [(m(I_r) - 1) \left\lceil \frac{2^r}{x \log(K)} \right\rceil]$. Notice that one group may have less than $\lceil \frac{2^r}{x \log(K)} \rceil$ items. This implies that

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

Using $\lceil a \rceil \geq a$ for $a \geq 0$, $m(I_r) \leq 2x \log(K) size(I_r) + 1$. For $x \geq 1$ and $K \geq 2$, we have $r(0) > \log(x) \geq 0$ and, therefore, $r(0) \geq 1$. Since the number of intervals for the medium items is at most $r(1) - r(0) + 1 \leq r(1) \leq \lfloor \log(yK) \rfloor$, the total number of rounded medium sizes $\sum_{r \geq r(0)} m(I_r) \leq \sum_{r \geq r(0)} (2x \log(K) size(I_r) + 1) \leq 2x \log(K) \sum_{r \geq r(0)} size(I_r) + \log(yK)$. Since all medium items fit into $K$ bins and $x, y$ are constants, $size(I_{medium}) = \sum_{r \geq r(0)} size(I_r) \leq K$ and

$$\sum_{r \geq r(0)} m(I_r) \leq 2xK \log(K) + \log(yK) \leq O(K \log(K)).$$

□

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 at most $R \leq O(K \log(K))$ 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 $x_j \geq \frac{1}{yK}$ 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 $\sum_j p(j)x_j + \mathit{preass}(b) \leq 1$ where $\mathit{preass}(b)$ 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

$$2^{O(K \log^2(K))} \cdot 2^{O(K \log^2(K))} \cdot \mathit{poly}(K) \leq 2^{O(K \log^2(K))}.$$

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 $\bigcup_{r \geq r(0)} J_r$. Notice that the other set $\bigcup_{r \geq r(0)} J'_r$ 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 at most $O(n)$ 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 $I$ into three groups $I_{large}$, $I_{medium}$, and $I_{small}$.
**(2)** Assign the large items to $K$ bins considering all 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 $b_{K+1}$.
**(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 $\mathbf{v}$.

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 $i \in [k]$.

Lemma 5 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 5** **($\star$).** *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.*

### 3.1 Non-averaging and sumfree 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.* Such sets have already been studied by several researchers [1, 3]. Although, up to our knowledge, the construction presented here does not appear in the literature in this form, it applies only standard techniques[1].

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, $|\mathcal{X}| \geq n$.

Lemmas 6 and 7 show that we can easily construct a non-averaging set $\mathcal{A}$ from $\mathcal{X}$, having $n$ elements. Setting $d = 2$, we get 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.

**Lemma 6 ($\star$).** *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 $\mathbf{u}_1 = \mathbf{u}_2 = \cdots = \mathbf{u}_k = \mathbf{v}$.*

**Lemma 7 ($\star$).** *If $b = k(m-1)+1$, then the set $\mathcal{A} = \{v^1 + v^2 b + \cdots + v^c b^{c-1} \mid \mathbf{v} \in \mathcal{X}\}$ 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 $O(2^d nN)$.*

A set $\mathcal{F}$ is *$k$-sumfree*, if for any two sets $S_1, S_2 \subseteq \mathcal{F}$ of the same size $k' \leq k$, $\sum_{x \in S_1} x = \sum_{x \in S_2} x$ holds if and only if $S_1 = S_2$. Such sets have been studied extensively in the literature, also under the name $B_k$-sequences [8, 9, 15].

It is easy to verify that the set $S = \{(k+1)^i \mid 0 \leq i < n\}$ is a $k$-sumfree set of size $n$. The maximal element in such a set is of course $(k+1)^{n-1}$. Intuitively, 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. Since no vector in $\mathbb{N}^n$ can be obtained in more than one different ways as the sum of at most $k$ vectors from $V_S$, an easy argument shows that $S$ is $k$-sumfree.

Although this will be sufficient for our purposes, we mention that a construction due to Bose and Chowla [2] shows that a $k$-sumfree set of size $n$ with maximum element at most $(2n)^k$ can also be found (see also [9], 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 8 ([2]).** *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 5, it immediately implies Theorem 2.

**Lemma 9.** 10-UNARY VECTOR BIN PACKING *is* W[1]*-hard.*

*Proof.* We present an FPT reduction from the W[1]-hard CLIQUE parameterized by the size of the desired clique. Let $G = (V, E)$ and $k$ be the input graph and the parameter given for CLIQUE. We assume $V = [n]$ and $|E| = m$, and we write

[1] We thank Imre Ruzsa for explaining us these techniques.

$x_h y_h$ for the $h$-th edge of $G$ according to some arbitrary ordering. We construct an instance $\mathcal{I}$ of 10-Unary Vector Bin Packing with $\binom{k}{2}+k+1$ bins.

**Item sizes.** The sizes of the items in $\mathcal{I}$ will be contained in $S \cup T$, where $S = \bigcup_{(i,j)\in\binom{[k]}{2}} S_{i,j}$, $T = T^{=} \cup T^{<} \cup T^{>}$, $T^{=} = \bigcup_{i\in[k]} T_{i,i}$, $T^{<} = \bigcup_{(i,j)\in\binom{[k]}{2}} T_{i,j}$, and $T^{>} = \bigcup_{(j,i)\in\binom{[k]}{2}} T_{i,j}$; here we use $\binom{[k]}{2} = \{(i,j) \mid 1 \le i < j \le k\}$. For each possible pair of $i$ and $j$, $|S_{i,j}| = m$ and $|T_{i,j}| = n$ will hold.

To determine the item sizes, we first construct a $k$-non-averaging set $\mathcal{A}$ of size $n$, using Lemma 7. Let $\mathcal{A}$ contain the elements $a_1 \le a_2 \le \cdots \le a_n$. By Lemma 7, we know $a_n = O(k^2n^2)$. Let $A = \sum_{h\in[n]} a_h$, clearly $A = O(k^2n^3)$.

We also construct a $k$-sumfree set $\mathcal{F}$ containing $k^2$ elements, using Lemma 8. Let us index the elements of $\mathcal{F}$ by pairs of the form $(i,j) \in [k]^2$, so let $\mathcal{F} = \{f_{i,j} \mid (i,j) \in [k]^2\}$. We also assume that $f_{i_1,j_1} < f_{i_2,j_2}$ holds if and only if $(i_1,j_1)$ is lexicographically smaller than $(i_2,j_2)$. By Lemma 8, we know $f_{k,k} = O(k^{2k})$. Again, we let $F = \sum_{f\in\mathcal{F}} f$, so $F = O(k^{2k+2})$.

For some $1 \le i < j \le k$, let $S_{i,j} = \bigcup_{h=1}^{m} \mathbf{s}_{i,j}(h)$, and for some $1 \le i,j \le k$ let $T_{i,j} = \bigcup_{h=1}^{n} \mathbf{t}_{i,j}(h)$. The exact values of $\mathbf{s}_{i,j}(h)$ and $\mathbf{t}_{i,j}(h)$ are as follows:

$$\begin{aligned}
\mathbf{s}_{i,j}(h) &= (ik+j,1,0,0,0,0,0,0,a_{x_h},a_{y_h}) && \text{if } (i,j) \in \tbinom{[k]}{2},\ h \in [m],\\
\mathbf{t}_{i,i}(h) &= (0,0,f_{i,i},1,0,0,0,0,(k-i)a_h,(i-1)a_h) && \text{if } i \in [k],\ h \in [n],\\
\mathbf{t}_{i,j}(h) &= (0,0,0,0,f_{i,j},1,0,0,a_h,0) && \text{if } (i,j) \in \tbinom{[k]}{2},\ h \in [n],\\
\mathbf{t}_{i,j}(h) &= (0,0,0,0,0,0,f_{i,j},1,0,a_h) && \text{if } (j,i) \in \tbinom{[k]}{2},\ h \in [n].
\end{aligned}$$

**Bin capacities.** We define $\binom{k}{2}+k+1$ bins as follows: we introduce bins $\mathbf{p}_{i,j}$ for each $(i,j) \in \binom{[k]}{2}$, bins $\mathbf{q}_i$ for each $i \in [k]$, 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 $j$). To define $\mathbf{q}_i$, we let $F_i^{<} = \sum_{i<j\le k} f_{i,j}$ and $F_i^{>} = \sum_{1\le j<i} f_{i,j}$ for each $i \in [k]$. Finally, we set the capacity of $\mathbf{r}$ in a way that the total size of the bins equals the total size of the items. Hence, any solution must completely fill all bins.

$$\begin{aligned}
\mathbf{p}_{i,j} &= (ik+j,1,0,0,(n-1)f_{i,j},n-1,(n-1)f_{j,i},n-1,A,A)\\
\mathbf{q}_i &= (0,0,(n-1)f_{i,i},n-1,F_i^{<},k-i,F_i^{>},i-1,(k-i)A,(i-1)A)
\end{aligned}$$

It is easy to see that $\mathbf{r} \in \mathbb{N}^{10}$. Observe that $|S \cup T| = m\binom{k}{2} + nk^2$, the unary encoding of the item sizes in $S$ needs a total of at most $O(mk^4 + nk^2A)$ bits, and the unary encoding of the item sizes in $T$ needs a total of at most $O(nF + k^3A)$ bits. By the bounds on $A$ and $F$, 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 some item $\mathbf{t}_{i,i}(h)$. Choosing an index $h \in [n]$ for each $i$ will correspond to choosing $k$ vertices from $G$. Next, we have to fill up the bin $\mathbf{q}_i$, by taking altogether $k-1$ items from $T^{<}$ and $T^{>}$ in a way such that the sum of their last two coordinates equals the last two coordinates of $\mathbf{t}_{i,i}(h)$. The sumfreeness of $\mathcal{F}$ and the non-averaging property of $\mathcal{A}$ will imply that the chosen items must be of the form $\mathbf{t}_{i,j}(h)$ and $\mathbf{t}_{j,i}(h)$ 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 $\mathbf{s}_{j,i}(e)$ will only do if the edge corresponding to $e \in [m]$ connects the vertices corresponding to $h_a$ and $h_b$, ensuring that the chosen vertices form a clique.

**Correctness.** Now, let us show formally that $\mathcal{I}$ is solvable if and only if $G$ has a clique of size $k$. 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 $\{P_{i,j} \mid (i,j) \in \binom{[k]}{2}\}$, $\{Q_i \mid i \in [k]\}$, and $R$ such that

$$\mathbf{p}_{i,j} = \sum_{\mathbf{v} \in P_{i,j}} \mathbf{v} \quad \text{for each } (i,j) \in \tbinom{[k]}{2}, \tag{1}$$

$$\mathbf{q}_i = \sum_{\mathbf{v} \in Q_i} \mathbf{v} \quad \text{for each } i \in [k], \text{ and} \tag{2}$$

$$\mathbf{r} = \sum_{\mathbf{v} \in R} \mathbf{v}. \tag{3}$$

**Direction $\Rightarrow$.** First, we argue that if $G$ has a clique of size $k$, then $\mathcal{I}$ is solvable. Suppose that $c_1, c_2, \dots, c_k$ form a clique in $G$. Let $d_{i,j}$ be the number for which $c_i c_j$ is the $d_{i,j}$-th edge of $G$. Using this, we set $P_{i,j}$ for each $(i,j) \in \binom{[k]}{2}$ and $Q_i$ for each $i \in [k]$ as follows, letting $R$ include all the remaining items.

$$P_{i,j} = \{\mathbf{t}_{i,j}(h) \mid h \neq c_i\} \cup \{\mathbf{t}_{j,i}(h) \mid h \neq c_j\} \cup \{\mathbf{s}_{i,h}(d_{i,j})\}. \tag{4}$$

$$Q_i = \{\mathbf{t}_{i,j}(c_i) \mid j \neq i\} \cup \{\mathbf{t}_{i,i}(h) \mid h \neq c_i\}. \tag{5}$$

It is easy to see that the sets $P_{i,j}$ for some $(i,j) \in \binom{[k]}{2}$ and the sets $Q_i$ for some $i \in [k]$ are all pairwise disjoint. Thus, in order to verify that this indeed yields a solution, it suffices to check that (1) and (2) hold, since in that case, (3) follows from the way $\mathbf{r}$ is defined. For any $(i,j) \in \binom{[k]}{2}$, using

$$\begin{aligned}
\sum_{h \neq c_i} \mathbf{t}_{i,j}(h) &= \sum_{h \neq c_i} (0,0,0,0,f_{i,j},1,0,0,a_h,0) \\
&= (0,0,0,0,(n-1)f_{i,j},n-1,0,0,A-a_{c_i},0), \\
\sum_{h \neq c_j} \mathbf{t}_{j,i}(h) &= \sum_{h \neq c_j} (0,0,0,0,0,0,f_{j,i},1,0,a_h) \\
&= (0,0,0,0,0,0,(n-1)f_{i,j},n-1,0,A-a_{c_j}), \\
\mathbf{s}_{i,h}(d_{i,j}) &= (ik+j,1,0,0,0,0,0,0,a_{c_i},a_{c_j}),
\end{aligned}$$

we get (1) by the definition of $P_{i,j}$. To see (2), we only have to use the definition of $Q_i$, and sum up the equations below:

$$\sum_{i<j\leq k} \mathbf{t}_{i,j}(c_i) = \sum_{i<j\leq k} (0,0,0,0,f_{i,j},1,0,0,a_{c_i},0)$$

$$= (0,0,0,0,F_i^{<},k-i,0,0,(k-i)a_{c_i},0),$$

$$\sum_{1\le j<i} \mathbf{t}_{i,j}(c_i) = \sum_{1\le j<i} (0,0,0,0,0,0,f_{i,j},1,0,a_{c_i})$$

$$= (0,0,0,0,0,0,F_i^{>},i-1,0,(i-1)a_{c_i}),$$

$$\sum_{h\neq c_i} \mathbf{t}_{i,i}(h) = \sum_{h\neq c_i} (0,0,f_{i,i},1,0,0,0,0,(k-i)a_h,(i-1)a_h)$$

$$= (0,0,(n-1)f_{i,i},n-1,0,0,0,0,(k-i)(A-a_{c_i}),(i-1)(A-a_{c_i})).$$

**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 (1), (2), and (3). We show that this implies a clique of size $k$ in $G$.

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 bin $\mathbf{x}$, we can immediately count the number of items from $S$, $T^{=}$, $T^{<}$, and $T^{>}$ that are contained in $X$. The following table shows the information obtained by this argument for each possible $X$.

| | $\lvert X\cap S\rvert$ | $\lvert X\cap T^{=}\rvert$ | $\lvert X\cap T^{<}\rvert$ | $\lvert X\cap T^{>}\rvert$ |
|---|---|---|---|---|
| $X = P_{i,j}$ for some $(i,j)$ | $1$ | $0$ | $n-1$ | $n-1$ |
| $X = Q_i$ for some $i$ | $0$ | $n-1$ | $k-i$ | $i-1$ |
| $X = R$ | $(m-1)\binom{k}{2}$ | $k$ | $0$ | $0$ |

Next, observe that $r^3 = \sum_{i\in[k]} f_{i,i}$. This means that $R$ contains exactly $k$ vectors from $\bigcup_{i\in[k]} T_{i,i}$ such that the third coordinate of their sum is $\sum_{i\in[k]} f_{i,i}$. But since $\mathcal{F}$ is $k$-sumfree, this can only happen if $R$ contains exactly one vector from each of $T_{1,1}, T_{2,2}, \dots, T_{k,k}$. Let these vectors be $\{\mathbf{t}_{i,i}(c_i) \mid i \in [k]\}$. We claim that the vertices $\{c_i \mid i\in[k]\}$ form a clique in $G$.

Using $q_i^3 = (n-1)f_{i,i}$, the table above, and $f_{1,1} < f_{2,2} < \cdots < f_{k,k}$ we obtain that $Q_i$ must contain every item in $T_{i,i} \setminus \{\mathbf{t}_{i,i}(c_i)\}$, for each $i\in[k]$. Also, we know that $Q_i$ must contain $k-i$ items from $T^{<}$ and $i-1$ from $T^{>}$, so from the values of $q_i^5$ and $q_i^7$ and the fact that $\mathcal{F}$ is $k$-sumfree, we also obtain that $Q_i$ must contain exactly one item from each of the sets $T_{i,j}$ where $j\neq i$. Note that apart from these $(n-1)+(k-1)$ vectors, $Q_i$ cannot contain any other items.

Now, note that the last two coordinates of the sum $\sum_{h\neq c_i} \mathbf{t}_{i,i}(h)$ are $(k-i)(A-a_{c_i})$ and $(i-1)(A-a_{c_i})$. Since the last two coordinates of $\mathbf{q}_i$ are $(k-i)A$ and $(i-1)A$, we get that $\sum_{\mathbf{v}\in Q_i\setminus T_{i,i}} \mathbf{v}$ must have $(k-i)a_{c_i}$ and $(i-1)a_{c_i}$ 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 $j\neq i$. Fixing some $i$ and letting $T_{i,j}\cap Q_i = \{t_{i,i}(h_j)\}$, this implies $\sum_{i<j\le k} a_{h_j} = (k-i)a_{c_i}$ and $\sum_{1\le j<i} a_{h_j} = (i-1)a_{c_i}$. But as $\mathcal{A}$ is $k$-non-averaging, this yields $h_j = c_i$ for each $j\neq i$. This means that (5) holds.

Next, let us consider the set $P_{i,j}$ for some $(i,j)\in\binom{[k]}{2}$. 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 $d_{i,j}$ such that $P_{i,j}\cap S_{i,j} = \mathbf{s}_{i,j}(d_{i,j})$. Furthermore, the table above and the result (5) shows that $P_{i,j}$ must contain $(n-1)$ items from both of the sets $T^{<}$ and $T^{>}$. Recall that $\{\mathbf{t}_{i,j}(c_i) \mid (i,j) \in [k]^2\} \subseteq \bigcup_{i\in[k]} Q_i$. Using $p^5_{i,j} = (n-1)f_{i,j}$ and $p^7_{i,j} = (n-1)f_{j,i}$, and taking into account the ordering of the elements of $\mathcal{F}$, it follows that (4) holds.

Finally, let us focus on the last two coordinates of the sum $\sum_{\mathbf{v}\in P_{i,j}} \mathbf{v}$. Clearly, if $i < j$ then 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, (4) and the definition of $\mathbf{p}_{i,j}$ yield that $\mathbf{s}_{i,j}(d_{i,j})$ 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. □

## References

1. N. Alon and I. Z. Ruzsa. Non-averaging subsets and non-vanishing transversals. *J. Comb. Theory, Ser. A*, 86(1):1–13, 1999.
2. R. C. Bose and S. Chowla. Theorems in the additive theory of numbers. *Comment. Math. Helv.*, 37(1):141–147, 1962-63.
3. A. P. Bosznay. On the lower estimation of non-averaging sets. *Acta Math. Hung.*, 53:155–157, 1989.
4. W. F. de la Vega and G. Lueker. Bin packing can be solved in within $1+\epsilon$ in linear time. *Combinatorica*, 1:349–355, 1981.
5. J. E. G. Coffman, M. R. Garey, and D. S. Johnson. Approximation algorithms for bin packing: A survey. In D. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*, pages 46–93. PWS Publishing, Boston, 1997.
6. F. Eisenbrand and G. Shmonin. Caratheodory bounds for integer cones. *OR Letters*, 34:564–568, 2006.
7. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.
8. 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.
9. H. Halberstam and K. F. Roth. *Sequences.* Springer-Verlag, New York, 1983.
10. 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.
11. R. Kannan. Minkowski's convex body theorem and integer programming. *Math. of OR*, 12:415–440, 1987.
12. N. Karmarkar and R. 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.
13. H. Lenstra. Integer programming with a fixed number of variables. *Math. of OR*, 8:538–548, 1983.
14. S. Plotkin, D. Tardos, and E. Tardos. Fast approximation algorithms for fractional packing and covering problems. *Math. of OR*, 20:257–301, 1995.
15. I. Z. Ruzsa. Solving a linear equation in a set of integers. I. *Acta Arith.*, 65(3):259–282, 1993.
16. D. Simchi-Levi. New worst-case results for the bin-packing problem. *Naval Res. Logist.*, 41(4):579–585, 1994.