Truly Asymptotic Lower Bounds for Online Vector Bin Packing

János Balogh
Institute of Informatics, University of Szeged, Hungary

Ilan Reuven Cohen
Faculty of Engineering, Bar-Ilan University, Ramat-Gan, Israel

Leah Epstein
Department of Mathematics, University of Haifa, Israel

Asaf Levin
Faculty of Industrial Engineering and Management, Technion, Haifa, Israel

Abstract

In this work, we consider online d-dimensional vector bin packing. It is known that no algorithm can have a competitive ratio of $o(d/\log^2 d)$ in the absolute sense, although upper bounds for this problem have always been presented in the asymptotic sense. Since variants of bin packing are traditionally studied with respect to the asymptotic measure, and since the two measures are different, we focus on the asymptotic measure and prove new lower bounds of the asymptotic competitive ratio. The existing lower bounds prior to this work were known to be smaller than 3, even for very large d. Here, we significantly improved on the best known lower bounds of the asymptotic competitive ratio (and as a byproduct, on the absolute competitive ratio) for online vector packing of vectors with $d \geq 3$ dimensions, for every dimension d. To obtain these results, we use several different constructions, one of which is an adaptive construction with a lower bound of $\Omega(\sqrt{d})$. Our main result is that the lower bound of $\Omega(d/\log^2 d)$ on the competitive ratio holds also in the asymptotic sense. This result holds also against randomized algorithms, and requires a careful adaptation of constructions for online coloring, rather than simple black-box reductions.

2012 ACM Subject Classification Mathematics of computing → Approximation algorithms; Mathematics of computing

Keywords and phrases Bin packing, online algorithms, approximation algorithms, vector packing

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2021.8

Category APPROX

Related Version *Previous Version*: https://arxiv.org/abs/2008.00811
Previous Version: https://arxiv.org/abs/2007.15709

Funding The research of J. Balogh was supported by the project "Extending the activities of the HU-MATHS-IN Hungarian Industrial and Innovation Mathematical Service Network" EFOP-3.6.2-16-2017-00015, and the project "Integrated program for training new generation of scientists in the fields of computer science," EFOP-3.6.3-VEKOP-16-2017-00002, supported by the European Union and co-funded by the European Social Fund. The research of A. Levin was partially supported by ISF - Israeli Science Foundation grant number 308/18.

Acknowledgements The results of this paper are based on the arxiv versions [9, 11]. Part of the work in [11] has been done while Ilan Reuven Cohen was a postdoctoral fellow at CWI Amsterdam and TU Eindhoven, he would like to thank Nikhil Bansal for discussions and suggestions related to fractional coloring and ideas from [27].

1 Introduction

We study the classical vector packing problem (VP) [21, 20, 2, 1, 3]. In VP with dimension $d \geq 2$, a set of items is given, where every item is a non-zero d-dimensional vector, whose components are rational numbers in $[0, 1]$. This set is to be partitioned into subsets called bins, such that the vector sum of every subset does not exceed 1 in any component. Here, we consider lower bounds on the worst-case performance guarantees of online algorithms for VP. Online algorithms obtain input items one by one, and pack each new item irrevocably before the next item is presented, into an empty (new), or non-empty bin. Such algorithms receive an input as a sequence, while offline algorithms receive an input as a set. An arbitrary optimal offline algorithm, which uses the minimum number of bins for packing the items of the input or instance, is denoted by OPT. For an input L and algorithm A, we let $A(L)$ denote the number of bins that A uses to pack L, also termed the *cost of algorithm A on input L*. For a randomized algorithm we denote the expected value of the cost of the algorithm A on input L as $\mathbb{E}[A(L)]$. We also let $OPT(L)$ denote the number of bins that OPT uses for a given input L. The absolute competitive ratio of an algorithm A is defined as the supremum ratio over the following ratios. These are the ratios for all inputs L, where the ratio for L is the ratio between the number of its bins $A(L)$ and $OPT(L)$, the number of the bins of OPT. The asymptotic competitive ratio is the limit of absolute competitive ratios R_K when K tends to infinity, and R_K takes into account only inputs for which OPT uses at least K bins. That is, the asymptotic competitive ratio of A is:

$$\lim_{K \to \infty} \sup_{OPT(L) \geq K} \frac{A(L)}{OPT(L)}.$$

For randomized online algorithms the definition for the asymptotic competitive ratio and for the absolute competitive ratio is the same, except that we consider $\mathbb{E}[A(L)]$ instead of $A(L)$. In this paper, we focus mostly on the asymptotic competitive ratio, which is the most natural measure for bin packing algorithms, and we sometimes refer to it by the term *competitive ratio*. When we discuss the absolute competitive ratio, we use this last term explicitly.

Note that VP is defined as a distinct optimization problem for every fixed value of d. For each such value (which is a dimension), there might be an online algorithm that is the best possible with respect to the absolute competitive ratio (the algorithm whose absolute competitive ratio is minimized), and there might be an online algorithm that is the best possible with respect to the asymptotic competitive ratio. These algorithms may be different. Denote by $\textsc{Abs}(d)$ the best possible absolute competitive ratio of a deterministic online algorithm for VP with dimension d, and let $\textsc{Asym}(d)$ denote the best possible asymptotic competitive ratio of a deterministic online algorithm for VP with dimension d. If there are values of d for which such best possible online algorithms do not exist, we define the corresponding values of $\textsc{Abs}(d)$ and $\textsc{Asym}(d)$ as the infimums of the corresponding ratios, where the infimum is taken over all online algorithms for VP with d dimensions. Thus, these values are well-defined for all d even though we currently do not know their values. It is known that these values are mostly linear in d (see below).

Observe that both $\textsc{Abs}(d)$ and $\textsc{Asym}(d)$ are monotone non-decreasing functions of d, as we can use the online algorithm with larger dimension in order to pack the lower dimension vectors by simulating a higher dimension input using additional components, which are always set to 0. Furthermore, by the definitions of the asymptotic competitive ratio and the absolute competitive ratio, we conclude that for every d, we have $\textsc{Asym}(d) \leq \textsc{Abs}(d)$.

In this work, we improve on the known lower bounds of the asymptotic competitive ratio for all fixed values of d that are at least 3 for online VP. We focus mostly on deterministic

online algorithms, while our main result uses an oblivious adversary and holds also against randomized online algorithm. This also improves the known results for the absolute ratios, that is, we improve upon the state of the art of lower bounding $\textsc{Asym}(d)$ for all $d \geq 3$. Our main result is that the order of growth of the asymptotic competitive ratio is $\Omega(\frac{d}{(\log d)^2})$. Recall that the term asymptotic here, does not refer to the asymptotic growth of d, but to the asymptotic definition of the competitive ratio, which is the most common measure for bin packing problems. To do that, we present four constructions leading to different lower bounds on $\textsc{Asym}(d)$. For every specific value of d, one may use the construction that leads to the largest possible lower bound. In Section 2, we present our results for very large values of d. That is, we show that for $d > 16$, there is a lower bound of $\frac{d-1}{8(\log_2 d)^3}$ on $\textsc{Asym}(d)$, and for sufficiently large and fixed values of d, the lower bound determined for the asymptotic competitive ratio for online VP is $\frac{d}{2^{11}\cdot(\log_2 d)^2}$. The lower bounds presented in Section 2 are based on an oblivious adversary. An elaborate explanation for using values of d that are sufficiently large is also provided in Section 2.

Next, we present the improved lower bounds for relatively small values of d against deterministic online algorithms. In Section 3, we show a lower bound of at least $\frac{\lfloor\sqrt{d-1}\rfloor+2}{2}$ that applies for all fixed values of $d \geq 2$. For example, in the case $d = 14$ we obtain a lower bound of 2.5 on the asymptotic competitive ratio. Then, in Section 4 we show a lower bound of $\frac{9}{4} = 2.25$ on $\textsc{Asym}(3)$. Finally, in Section 5, we introduce a lower bound of $\frac{76}{29} \approx 2.62$ on $\textsc{Asym}(8)$. The last three constructions, presented in Sections 3, 4, and 5, are based on a technique called adaptive constructions for packing problems, explained in detail below. We note that our lower bound of $\frac{\lfloor\sqrt{d-1}\rfloor+2}{2}$ is the largest lower bound that we establish for a large variety of values of d. This holds even for extremely large values of d like $d = 2^{30}$, for which the value of this lower bound is at least $2^{14} > 16000$. Comparing this result to the values of lower bounds presented in Section 2, we find that they are much smaller, and in fact the values resulting from the constructions of that section are not larger than 4972 and 583. Thus, for any reasonable value of d, the lower bounds obtained based on the adaptive construction method are much better than the ones in Section 2.

The lower bounds on $\textsc{Abs}(d)$ established in Azar et al. [2] were not computed explicitly, in the sense that they are only stated as $\Omega(d^{1-\varepsilon})$ for every $\varepsilon > 0$. These lower bounds hold only in the absolute sense, and their order of growth is $\Omega\left(\frac{d}{(\log_2 d)^2}\right)$. In order to compare their bounds to our lower bounds on $\textsc{Asym}(d)$, we examined their proofs. Their lower bounds are basically the deterministic lower bounds on node coloring of graphs where the nodes arrive in an online fashion, thus they are using black-box reductions from the (deterministic) lower bounds of Halldórsson and Szegedy [22]. These black-box reductions are the main reason that lower bounds cannot be obtained on the asymptotic competitive ratio, but only on the absolute competitive ratio, which may be larger (and it is not the standard tool to analyze bin packing problems). Note that we improve the known absolute lower bound for a large number of values of d. For example, the deterministic lower bound resulting from the construction of [22] for $d = 60$ is 3.75, while our results imply a lower bound above 4.8 (which holds even in the asymptotic case, and for a slightly lower dimension of 57).

In the work of [22], the part of randomized lower bounds consists of two constructions. In the first one, the value of d is relatively large, thus we can use Stirling's formula to approximate $\lceil\log_2 d\rceil!$ with a constant multiplicative error, while the second one holds for all dimensions d. In Section 2, we use the ideas of Halldórsson and Szegedy [22] to obtain similar lower bounds on the asymptotic competitive ratio of VP. Unlike the work of [2], we do not apply black-box reduction from node coloring. The main motivation for that is that we need a lower bound with respect to a different coloring problem in which we are interested

in *fractional coloring*. Since this variant was not studied before, we cannot use an existing result for it, nor can we use a black-box reduction from it. Instead, we present the lower bound construction for VP. The transition to this fractional coloring is the main tool that we use in order to lift the construction of [2], so that it will allow us to prove a lower bound on the asymptotic competitive ratio for VP. We refer to [11] for a different approach that is based on a black-box reduction from fractional coloring to randomized coloring together with a black-box reduction from the randomized lower bound of [22]. This alternative approach of [11] leads to inferior lower bounds, therefore, we base our result on the approach of [9] instead.

Next, we survey the literature for the bounds of $\textsc{Asym}(d)$. The special case of $d = 1$ of VP is simply the bin packing problem (BP), which has been studied for half a century [23, 24]. The current best bounds on $\textsc{Asym}(1)$ (for the online version) are a lower bound of 1.54278 [7] and an upper bound of 1.57829 [5], while $\textsc{Abs}(1) = \frac{5}{3}$ [8]. In [4, 6], the authors considered a generalization of BP, known as the Cardinality Constrained Bin Packing (CCBP). CCBP is defined as follows; each item has a scalar size in $[0, 1]$ (one-dimensional), and in addition , the constraint that the sum of items sizes in each bin does not exceed 1, there is an integer parameter $k \geq 2$, such that no bin can have more than k items. Note that CCBP is a special case of VP with $d \geq 2$, by defining a vector for each item, where its first component is the item's size and its second component is $\frac{1}{k}$ (the other components are zeros). In [6], the authors consider the case of VP with $d = 2$, and present its difference from CCBP. They demonstrated that $\textsc{Asym}(2) \geq 2.03731129$ with respect to VP in two dimensions, and thus for all $d \geq 2$ it was established that $\textsc{Asym}(d) \geq 2.03731129$, whereas for CCBP there is a 2-competitive algorithm. In [3] it is shown that when d grows to infinity there is a lower bound that tends to e, thus $\lim_{d\to\infty} \textsc{Asym}(d) \geq e$. It is interesting to note that the lower bound in [3] applies even if all components of all vectors are small. That is, for every small $\varepsilon > 0$, where all components are smaller than ε, the lower bound of [3] holds, and it holds even when ε tends to 0. This matches the upper bound for the very specific case of VP (of small components) established by [1]. Prior to the publications of [3, 6], the lower bounds on $\textsc{Asym}(d)$ were weaker [20, 14, 13].

As for upper bounds on $\textsc{Asym}(d)$, the best known result [21] is (still) that the First-Fit algorithm has an asymptotic competitive ratio of $d + 0.7$ for VP in d dimensions. Prior to that work, there was a slightly weaker bound of $d+1$ established by Kou and Markowsky [26]. For the absolute competitive ratio, the resulting bound is also $d + O(1)$ [21, 26] (an upper bound of $O(d)$ follows from the simple property that for greedy algorithms, no two bins of the output have a sum of at most 1 in all components). With respect to the literature on VP regarding oblivious adversary, all known upper bounds are using deterministic algorithms, while the lower bound of [3] is the unique lower bound larger than 2 that holds also using oblivious adversary (by applying standard adaptation). The lower bound of [6] is stated only for deterministic algorithms, and the means for adjusting it for use of an oblivious adversary are unknown.

We stress that prior to our work it was unknown whether there is an online algorithm A for VP for any dimension d (or for sufficiently large dimensions) such that for every input L, its cost $A(L)$ satisfies $A(L) \leq 3 \cdot OPT(L) + d$ (where d is the dimension of the input). This is due to the fact that the known lower bound of e holds for any additive term, while the non-constant known lower bound uses only d items for dimension d.

Vast literature is available on the offline versions of all the problems mentioned above [16, 10, 12, 15, 18, 19, 25, 28]. In particular, similar lower bounds to those of [1] on the absolute approximation ratio for VP (under a certain standard complexity assumption, that

NP≠ZPP) were observed for the offline scenario [16, 12], also by reductions from coloring problems. The lower bounds are for the absolute measures, since for every dimension d, the number of items is simply d. The other bin packing problems discussed here (BP and CCBP) admit polynomial time asymptotic approximation schemes [15, 18, 19, 25], but one can show that unless P=NP, such schemes cannot exist for the absolute cases, and the absolute approximation ratio is at least 1.5. This can serve as evidence that lower bounds for the absolute approximation ratio or the absolute competitive ratio are not sufficient for the analysis of the asymptotic approximation ratio or the asymptotic competitive ratio. In fact, it is true also for online bin packing problems that the absolute measure is different from the asymptotic one. For example, for BP, the best possible absolute competitive ratio is $\frac{5}{3}$ [8], while the asymptotic competitive ratio is significantly smaller [5]. For CCBP, one example is the parameter $k = 4$, for which the best possible absolute competitive ratio is 2 [4, 6], while an improved asymptotic competitive ratio below 2 is known [17].

Omitted proofs will appear in the full version of this work.

2 The lower bound for large values of d

In this section we consider the cases of very large dimensions. We will prove a lower bound of $\Omega(\frac{d}{\log^2 d})$ for sufficiently large dimensions d. Our lower bounds are not smaller than the weaker results of Azar et al. [2], which were established only for the absolute competitive ratio measure (as a function of the dimension, for large enough values of d). Here, we consider the stronger measure of the asymptotic competitive ratio, and even prove these lower bounds for randomized algorithms.

The input's sequence presented by the adversary is constructed in d phases, where each phase consists of N identical items. The construction uses integer parameter ν (which will be chosen later as a function of the number of dimensions), and maintains ν classes and $\mathcal{S}$, a subset of the ν class sets, $\mathcal{S} \subseteq X = 2^{[\nu]} \setminus \emptyset$ where $[\nu] = \{1, 2, \ldots, \nu\}$, let $0 < \varepsilon \leq \frac{1}{Nd}$. The adversary examines the online algorithm's expected values, and associates a class and a set for each phase, which will determine the items' phase components. We denote $c_j \in [\nu]$, and $S_j \in \mathcal{S}$ as the class and the set of the phase j, respectively.

Formally, at the beginning of phase $j \in [d]$, the adversary examines the algorithm's expected assignment and chooses a set $S_j \in \mathcal{S}$. The adversary presents N identical items, where each item component is defined as follows: the items have 1 as their jth component, all components of indexes larger than j are 0, while a component of index $i < j$ is ε if $c_i \notin S_j$ and 0 otherwise. Note that the previous phases classes $c_1, \ldots c_{j-1}$ have been already set. Finally, after examining the expected assignment of these vectors the adversary associates a class $c_j \in S_j$ for the phase.

First, observe that the non zero components' values are either 1 or $\epsilon \leq \frac{1}{Nd}$. Since the number of items is Nd, a subset of the items cannot be assigned to the same bin, only if exists a component for which there is an item with a value of 1 and another item with a non-zero value. In addition, a construction of this form satisfies that any solution has a cost of at least N (and of at most Nd), regardless of its specific details. Therefore, by letting N grow to infinity by proving a lower bound for these instances a lower bound on the asymptotic competitive ratio follows. Next, consider a bin B with a fixed realization of the random bits of the online algorithm (at some point during the lower bound construction). We associate the subset $S(B) \subseteq [\nu]$ of classes with every such bin, where $S(B)$ contains the classes of items that are packed into B, that is, $\ell \in S(B)$ if there exists at least one phase j such that $\ell = c_j$ and an item of phase j is packed into B. The set $S(B)$, named the associated set of

B may be extended later. Intuitively, the adversary chooses a set S_j and a class c_j in each phase, in order to increase the (expected) cardinality of the associated set ($|S(B)|$) of the online algorithm bins. Then, by using a corresponding potential function, we will obtain the lower bound of the expected number of bins of the online algorithm. Full details of the adversary's choices are presented later. First, we introduce several properties, which hold for any choice of adversary. The following claim characterizes the possible associated set that an algorithm may pack the items of phase j.

▷ Claim 1. Any algorithm may pack an item of phase j in bin B, if and only if it does not contain another item of phase j, and before the item is added it holds that $S(B) \subseteq S_j$.

Proof. Two items of the phase j cannot fit into a common bin, since there is 1 in the j'th component of the items. The number of items is $N \cdot d$, hence a collection of items fit into a bin, if and only if, no item in the collection has an ε component when another item of the collection has a value of 1 at the same component.

Given a bin B, for any $\ell \in S(B)$ there exists at least one value j', such that an item of phase j' is assigned to B, and the class of phase j' is ℓ. By the construction definition, the j' component of this item (and the bin B) is 1, and if $\ell \notin S_j$ the j'th component of items of phase j is ε. Hence, if item of phase j can be assigned to bin B then $\ell \in S_j$. On the other hand, if $S(B) \subseteq S_j$ then for any component $i < j$ of B which is 1, we have $c_i \in S(B) \subseteq S_j$, and by definition the j'th phase vector value at component i is 0. ◁

We observe that there exists a solution which uses at most $\nu \cdot N$ bins.

▷ Claim 2. For any choice of c_j, S_j there exists an offline solution which uses at most $\nu \cdot N$ bins.

Proof. A solution that opens N bins for each class and assigns the phase's vector according to the class is a feasible assignment by Claim 1, since for a bin we use to pack in phase j we have $S(B) \subseteq \{c_j\}$ and $c_j \in S_j$ by the algorithm's definition. ◁

Next, we will present an important characterization of the set $\mathcal{S}$ necessary for the analysis. The set $\mathcal{S}$ in the construction will depend on a parameters pair (α, β), and will require that (α, β) would be *satisfiable*.

▶ **Definition 3.** *Given ν, (α, β) is satisfiable if there exists a set of subsets $\mathcal{S} \subseteq X$ such that $|\mathcal{S}| \geq \alpha$, and for every pair $S, S' \in \mathcal{S}$ such that $S \neq S'$ we have that their symmetric difference $S \triangle S'$ satisfies $|S \triangle S'| \geq \beta$.*

The next claim characterizes two satisfiable pairs, which will be used in our lower bound construction.

▷ Claim 4. For any ν, $(\alpha, \beta) = (2^\nu - 1, 1)$ is satisfiable, and if ν is sufficiently large then $(\alpha, \beta) = (2^{\nu/4}, 0.3\nu)$ is satisfiable.

Proof. For the first part, consider $\mathcal{S}$ to be the set X, which has $2^\nu - 1$ elements, as required. Each pair of distinct elements represents non-equal subsets of $[\nu]$, so that they differ by at least one element. We fix the value of β to 1, in this case.

The second part was proven by [22], which demonstrated that if we pick a random sub-collection of subsets of $[\nu]$ with $2^{\lceil \nu/4 \rceil}$ subsets, each consisting of exactly $\lceil \nu/2 \rceil$ elements of $[\nu]$ (chosen independently and uniformly at random), then with some positive probability (for large enough value of ν) each pair of these selected subsets satisfies the condition on their symmetric difference. Using the probabilistic method, they were able to prove our claim (deterministically) for large enough values of ν (that they have not specified). ◁

For a specific satisfiable pair (α, β) and their corresponding set $\mathcal{S}$, and for any associated set $S(B) \subseteq [\nu]$, we denote a subset in $S \in \mathcal{S}$ that *represents* B, (where S is represented by B) if $|S \triangle S(B)| \leq \frac{\beta}{5}$. We will prove that any set is represented by at most a single set in $\mathcal{S}$.

▷ Claim 5. If for every pair $S, S' \in \mathcal{S}$ such that $S \neq S'$ we have $|S \triangle S'| \geq \beta$ and $\beta \geq 1$, then any bin B is represented by at most one set $S \in \mathcal{S}$.

Proof. Assume for contradiction that a bin B is represented by two sets $S_1, S_2 \in \mathcal{S}$. Recall that $|S_1 \triangle S_2| \geq \beta$ (by assumption), but $|S(B) \triangle S_i| \leq \frac{\beta}{5}$ for $i = 1, 2$ (by the definition of representation). First, consider the elements of $S_1 \setminus S_2$. Some of these elements belong to $S(B)$, while other do not. Observe that $(S_1 \setminus S_2) \cap S(B) \subseteq S(B) \triangle S_2$ so $|(S_1 \setminus S_2) \cap S(B)| \leq \frac{\beta}{5}$. Since $(S_1 \setminus S_2) \setminus S(B) \subseteq S(B) \triangle S_1$, we conclude that $|(S_1 \setminus S_2) \setminus S(B)| \leq \frac{\beta}{5}$. Therefore, $|S_1 \setminus S_2| \leq \frac{2\beta}{5}$. Similarly (by changing the roles of S_1 and S_2) we conclude that $|S_2 \setminus S_1| \leq \frac{2\beta}{5}$. Thus, $|S_1 \triangle S_2| \leq \frac{4\beta}{5}$, contradicting our assumption about $\mathcal{S}$. ◁

We are ready to prove our main lemma, which achieves a lower bound on the expected number of bins opened. We will demonstrate that the adversary may choose a set and a class that will increase the cardinality of the associated sets of the algorithm's bins in every step, which we will capture by using a potential function. Our potential function is the expected value of the sum of $|S(B)|$ over all bins B of the algorithm. That is, $\Phi = \mathbb{E}[\sum_B |S(B)|]$. Observe that the expected value of the cost of the online algorithm is at most Φ and not smaller than $\frac{\Phi}{\nu}$, since for any B it holds that $|S(B)| \leq \nu$. Let $\gamma = \lceil \frac{\beta}{5} \rceil$, this parameter is chosen to ensure that each associated set of a bin is represented by at most one set $S \in \mathcal{S}$.

▶ **Lemma 6.** *For a satisfiable pair (α, β) there exists an adversary strategy, such that the expected number of bins opened by the online algorithm is at least* $\min\{\alpha \cdot N/2, d \cdot \frac{\gamma}{\nu^2} \cdot \frac{N}{2}\}$.

Proof. We will show that if the expected number of bins before phase j is less than $\alpha \cdot N/2$, then there exists S_j, c_j, which will increase the potential Φ by at least $\frac{\gamma}{\nu} \cdot \frac{N}{2}$. The lemma holds since we can perform d phases, and since the number of bins opened is at least $\frac{\Phi}{\nu}$.

Prior to phase j, we let $n(S)$ for a set $S \in \mathcal{S}$ be the expected value of the number of bins (of the online algorithm) that it represents. After the assignment of the j'th phase items, we denote by $\tilde{n}^j(S)$ for a set $S \in 2^{[\nu]}$ the expected number of bins associated with S and a j'th phase item assigned into it. Note that the adversary may use $n(S), \tilde{n}^j(S)$ when determining S_j, c_j, respectively, and that these values do not depend on the realization of the random bits used by the algorithm.

Determining S_j. We obtained $\sum_{S \in \mathcal{S}} n(S) \leq \alpha \cdot \frac{N}{2}$ through the assumption that the expected number of bins is less than $\alpha \cdot N/2$, and since every bin of the algorithm is represented by at most one set. The adversary sets $S_j \in \mathcal{S}$ such that $n(S_j)$ is below $\frac{N}{2}$, and the existence of S_j can be guaranteed by the pigeonhole principle.

Determining c_j. The algorithm assigns each item of phase j to a distinct bin; by linearity of expectation we have,

$$N = \sum_S \tilde{n}^j(S) = \sum_{|S \triangle S_j| < \gamma} \tilde{n}^j(S) + \sum_{|S \triangle S_j| \geq \gamma} \tilde{n}^j(S) \leq \frac{N}{2} + \sum_{|S \triangle S_j| \geq \gamma} \tilde{n}^j(S),$$

where the first inequality is obtained by our choice of S_j. Therefore, we have

$$N/2 \leq \sum_{|S \triangle S_j| \geq \gamma} \tilde{n}^j(S) \leq \sum_{\ell \in S_j} \frac{\sum_{S : \ell \notin S} \tilde{n}^j(S)}{\gamma},$$

where the second inequality follows, since $\tilde{n}^j(S) > 0$ only if $S \subseteq S_j$ by Claim 1, so the same S will appear at least γ times in the inner summation on the right hand side. The adversary set $c_j \in S_j$ such that $\sum_{S:c_j \notin S} \tilde{n}^j(S)$ is above $\frac{\gamma}{\nu} \cdot \frac{N}{2}$, the existence of c_j can be guaranteed by the pigeonhole principle since $|S_j| \leq \nu$. Note that the increase in the potential function is exactly $\sum_{S:c_j \notin S} \tilde{n}^j(S)$, as required. ◀

2.1 Assembling the pieces together

Recall that by Claim 2 the cost of the optimal solution is at most $\nu \cdot N$. By Lemma 6 any online algorithm will use at least $\min\{\alpha \cdot N/2, d \cdot \frac{\gamma}{\nu^2} \cdot \frac{N}{2}\}$ number of bins. This value is maximized if $d \cdot \frac{\gamma}{\nu^2} \cdot \frac{N}{2} \approx \frac{\alpha N}{2}$. Thus, we need a method for selecting ν, if the dimension d is given. We will use a value ν for which the corresponding pair (α, β) satisfies $d \geq \frac{\nu^2 \alpha}{\gamma}$ (where γ is determined by β), and consequently the resulting lower bound would be $\frac{\alpha}{2\nu}$.

First, consider the case where d is relatively small and we use $(\alpha, \beta) = (2^\nu - 1, 1)$ and thus $\gamma = 1$, and ν is an integer such that $\nu^2 \cdot (2^\nu - 1) \leq d$. It is sufficient to require that $\nu^2 \cdot 2^\nu \leq d$, that is satisfied by letting $\nu = \lfloor \log_2 d - 2 \log_2 \log_2 d \rfloor$, as for this choice $\nu^2 2^\nu \leq (\log_2 d)^2 \cdot \frac{d}{(\log_2 d)^2} = d$. The resulting lower bound is not smaller than

$$\frac{\alpha}{2\nu} \geq \frac{2^{\log_2 d - 2\log_2\log_2 d} - 1}{4 \cdot (\log_2 d - 2\log_2 \log_2 d)} \geq \frac{d-1}{8(\log_2 d)^3} \ ,$$

where the last inequality holds for $d > 16$, as for these values of d we have that $8 \log_2 \log_2 d < 4 \log_2 d$.

Next, consider the case where the dimension is higher, and we could use $(\alpha, \beta) = (2^{\nu/4}, 0.3\nu)$, and thus $\gamma = 0.06\nu$. We pick ν as the largest integer such that $\frac{\alpha\nu^2}{\gamma} \leq d$ that is, $\alpha\nu = \nu 2^{\nu/4} \leq 0.06 \cdot d$. Letting $\nu' = \frac{\nu}{4}$ we will require $\nu' \cdot 2^{\nu'} \leq 0.015d$. This condition is satisfied, e.g. for $\nu' = \lfloor \log_2 d - \log_2 \log_2 d - 7 \rfloor$, as for this choice of ν' we have

$$\nu' \cdot 2^{\nu'} \leq (\log_2 d) \cdot 2^{\log_2 d - \log_2 \log_2 d - 7} = (\log_2 d) \cdot \frac{d}{\log_2 d \cdot 2^7} = \frac{d}{2^7} \leq 0.015d$$

and ν' is an integer and thus also $\nu = 4\nu'$ is integer. The resulting lower bound is

$$\frac{\alpha}{2\nu} = \frac{2^{\nu'}}{8\nu'} \geq \frac{2^{\log_2 d - \log_2 \log_2 d - 8}}{8 \cdot (\log_2 d - \log_2 \log_2 d - 7)} \geq \frac{\frac{d}{\log_2 d}}{2^{11} \cdot \log_2 d} = \frac{d}{2^{11} \cdot (\log_2 d)^2},$$

that holds for large enough values of d.

Thus, we conclude the construction of this section by the following theorem.

▶ **Theorem 7.** *For every fixed dimension $d > 16$, there is a lower bound of*

$$\frac{d-1}{8(\log_2 d)^3} \ ,$$

on the asymptotic competitive ratio of online randomized algorithms for VP. For sufficiently large and fixed values of d the lower bound on the asymptotic competitive ratio is

$$\frac{d}{2^{11} \cdot (\log_2 d)^2} \ .$$

3 A lower bound for medium sized dimensions

Let α, $\beta \geq 2$ be the two positive integers such that $d \geq 2 + \alpha(\beta - 2)$. Next we show a lower bound of $\frac{\alpha \cdot \beta}{\alpha + \beta - 2}$ for the corresponding special case. Note that choosing the values $\alpha = \beta$ results in a lower bound of $\Omega(\sqrt{d})$ so for very large dimension this result is inferior to the general lower bound we considered earlier. However, the hidden constants in the Ω notation are smaller for the current construction leading to better lower bounds for medium sized dimensions.

Let $N > d$ be a large integer such that $\frac{N}{\alpha}$ is an integer. Our sequence of items may have up to N^3 items, and we let $\varepsilon < \frac{1}{N^3}$. We will use the adaptive construction method to generate a sequence of scalar values where a_i is the value associated with the ith item, such that all values are smaller than ε, and furthermore the following condition holds. If an item is large, then its value is at least N^3 times larger than the value of a small item. The logical condition that we will use to define small and large items is that a d-dimensional item is large if it is packed into an empty bin and otherwise it is small. We stress the property that every item of the construction will have a one-dimensional associated value, and we will explain how this value is used in the definition of the d-dimensional item.

During the adaptive construction, after packing the current item, there will be a value $\mu < \frac{1}{N} < \frac{1}{2}$. The value of μ may decrease (but cannot increase) after assigning an item. The value μ will satisfy the property that the value of every large item that appeared up to (and including) the current iteration is strictly larger than μ while for any subset of items S where S contains only small items that appear in the instance (both during the prefix and later on) or large items that appear later on in the input sequence, the total value of S is strictly smaller than μ. This is obtained by letting μ be the current upper bound on values of items that can still be either small or large at termination, and reducing the length of the interval of possible values in the adaptive construction by a multiplicative factor of N^3 after each item. This is done by using the scheme of [6], and setting the predefined constant mentioned earlier (k) to N^3, this ensure that all such subsets S, whose number of elements will be less than N^3 (as this will be a valid upper bound on the number of items in the entire construction), will satisfy the requirement. Note that, in the construction all the successive items to a large item must be at least N^3 smaller than this large item since those items could be eventually small.

Our construction will have β phases, and it will be useful to denote by μ_i the current value of μ at the end of phase i (i.e., after packing the last item of phase i and modifying the current interval according to the rules of the adaptive construction). Furthermore, phase i uses the value μ_{i-1}.

The first and last phase have special properties while the intermediate phases ($\beta - 2$ phases) are all similar. Each phase lasts until the first point in time in which the algorithm has opened N new bins during this phase. Thus, we will ensure that the total cost of the algorithm is $N \cdot \beta$. We also maintain an integer value π denoting the *current component* that is being dealt with, and it has a special role in the construction. We will show later that π will always be an index of a component (i.e., $\pi \leq d$), even though it is increased frequently. The value π is an index of a component such that items have a very big (and close to 1) component of this index. By increasing π, we change (and increase) the position of this very big component in the construction. This value is initially set as $\pi = 2$ (while the very first component has a special role during the first phase), and it increases gradually, each time by 1, and it never decreases, as items are being presented. We will see that the value of π never exceeds d, and during the presentation of items of intermediate phases it will hold that $\pi \leq d - 1$.

The first phase

We construct a sequence of items, where the first component of every vector is $\frac{1}{N}$ while every other component equals to the value of the current item. Note that this phase ends after at most N^2 items, since any phase ends after the algorithm used N new bins, every new bin can contain at most N items of this phase, and there are no bins of previous phases which can be used.

The $\beta - 2$ intermediate phases

Every such phase will contain at most $\alpha \cdot N$ items. We keep a counter j of the index of the phase, where j is initialized to 2. At the end of phase $j - 1 \geq 1$ we set μ_{j-1} as we described above and we start presenting new items of the jth phase.

Each item of these $\beta - 2$ intermediate phases will consist of the following components. All components with indexes smaller than π are equal to 0, the current component with index π is set to $1 - \mu_{j-1} > \frac{1}{2}$, and all other components (of larger indexes) are equal to the value of the current item (of the adaptive construction). Recall that a new phase starts whenever the number of new bins during the current phase is N, and just before starting a new phase we also increase π by 1 (for $j = 2$, π is not increased but it is initialized). However, there are other events where we decide to increase the value of π by 1. These additional events are stated as follows. Whenever the number of large items (according to the adaptive construction) that were packed while the value of π is its current value, is $\frac{N}{\alpha}$, we increase the value of π by 1. This is done since the number of large items whose πth component is very big is the maximum possible number. We will show that an increase in the value of π will happen after at most N consecutive items for which we used the same value of π. This happens either due to the latter rule or due to the end of the phase (since π is always increased due to that event). Before presenting the vectors of the last phase, we prove the main correctness claims regarding the intermediate phases that allow our construction to have the required structure and allow us to prove the claimed lower bound on the asymptotic competitive ratio.

▶ **Lemma 8.** *The items of phase j (where $2 \leq j \leq \beta - 1$) cannot be packed into bins that were opened in an earlier phase, that is, bins used first for an item of an earlier phase. Additionally, the value of π remains constant without being increased for at most N items.*

▶ **Lemma 9.** *During an intermediate phase, the value of π is increased at most α times by* 1 *(including the increase due to the end of the phase).*

We consider the value of π at the beginning of the last phase. By the last lemma, we conclude that just before the moment when phase $\beta - 1$ ends (the last intermediate phase), we have $\pi \leq 2 + (\beta - 2) \cdot \alpha + (\beta - 1) \leq d - 1$ and π is increased to a value of at most d once that phase ends. This final value (of at most d) for π was not used as the value of π in the definition of items of any phase (after the very last time that π was increased, no items are defined so it was not used to define an item).

The last phase

In the last phase we present exactly N identical items that are defined as follows. In component d they are equal to $1 - \mu_{\beta-1}$ and all other components are 0. Observe that every bin that the algorithm has opened in one of the earlier phases has one large item whose dth component is larger than $\mu_{\beta-1}$, and thus the algorithm needs to open N new bins for these N items of the last phase.

Proving the resulting lower bound

Since there are β phases and the algorithm is forced to open N new bins in every phase, we conclude that the cost of the algorithm is exactly $N \cdot \beta$. Since N could be an arbitrary large integer, in order to prove the lower bound on the asymptotic competitive ratio of the algorithm, it suffices to show that the optimal offline cost is at most $N \cdot (1 + \frac{\beta-2}{\alpha}) + 1$. In order to present this proof, we will consider all items of the last phase as small items. We present an offline solution of cost at most $N \cdot (1 + \frac{\beta-2}{\alpha}) + 1$. This offline solution will pack all large items into $N \cdot (\frac{\beta-2}{\alpha}) + 1$ bins, and all small items into a disjoint set of N bins, and in total we will use at most $N \cdot (1 + \frac{\beta-2}{\alpha}) + 1$ bins. First, we consider the large items.

▶ **Lemma 10.** *There is an offline solution that packs all large items into at most* $N \cdot (\frac{\beta-2}{\alpha}) + 1$ *bins.*

Next, we consider packing of the small items into N bins.

▶ **Lemma 11.** *There is an offline solution that packs all small items into* N *bins.*

Thus, we conclude the following result.

▶ **Theorem 12.** *If* $d \geq \alpha(\beta - 2) + 2$*, then there is no online algorithm for VP whose asymptotic competitive ratio is smaller than* $\frac{\beta}{1+\frac{\beta-2}{\alpha}} = \frac{\alpha \cdot \beta}{\alpha+\beta-2}$.

For large values of d, we can use $\alpha = \beta = \lfloor \sqrt{d-1} \rfloor + 1 \geq \lceil \sqrt{d-1} \rceil$ for which $\alpha(\beta-2)+2 \leq (\sqrt{d-1}+1) \cdot (\sqrt{d-1}-1) + 2 = d - 1 - 1 + 2 = d$ and this lower bound on the asymptotic competitive ratio is $\frac{\alpha \cdot \beta}{\alpha+\beta-2} = \frac{\alpha^2}{2\alpha-2} = \frac{\alpha+1}{2} + \frac{1}{2(\alpha-1)} > \frac{\lfloor \sqrt{d-1} \rfloor}{2} + 1 \geq \frac{\sqrt{d-1}+1}{2} > \frac{\sqrt{d}}{2}$. However, for small dimensions we could do better. For example for $d = 98$, we could pick $\alpha = 12, \beta = 10$ and the lower bound on the asymptotic competitive ratio is $\frac{120}{20} = 6$ whereas using $\alpha = \beta = 10$ the lower bound is $\frac{100}{18} \approx 5.555$.

For small dimensions, namely $d = 6, 7, 9, 10$ and 11, the next estimation can be used. Letting $\beta = 3$ and $\alpha = d - 2$, the lower bound is greater or equal to $\frac{3\alpha}{\alpha+1} = \frac{3}{1+\frac{1}{\alpha}} = \frac{3}{1+\frac{1}{d-2}}$. It gives a lower bound of 2.4, 2.5, 2.625, 2.666, and 2.7 for the cases of $d = 6, 7, 9, 10, 11$, respectively. We mention several other small values of d. For $d = 12$, we can use $\alpha = 5$ and $\beta = 4$ to obtain a lower bound of $\frac{20}{7} \approx 2.857$. For $d = 14$, we can use $\alpha = 6$ and $\beta = 4$ to obtain a lower bound of 3. For $d = 16$, we can use $\alpha = 7$ and $\beta = 4$ to obtain a lower bound of $\frac{28}{9} \approx 3.111$. Improved bounds for the cases $d = 3, 4, 5, 8$ are presented in the next sections.

4 The case $d = 3$

Recall that the known lower bound on the asymptotic competitive ratio for $d = 2$ is just slightly above 2, and this was the best known constant lower bound for any small value of d till now. We prove here a lower bound of 2.25 for the case $d = 3$, and explain how it can be slightly improved.

We will use an adaptive construction as explained earlier. The construction is based on that of [6].

Let $K > 1000$ be a large integer, and let $\varepsilon > 0$ be a small constant (in particular, $\varepsilon < \frac{1}{K} < 0.001$). The input consists of three parts and we describe the parts one by one.

The first part of the input

Using an adaptive construction of values, we define a sequence of values in $(0, \varepsilon)$ such that any large value is strictly larger by a multiplicative factor larger than $10K$ from any small value. The binary condition is that the item is packed into an empty bin by the online algorithm.

Thus, an item packed into an empty bin is large, and otherwise the item is small. The number of items will be $2 \cdot K \cdot N$ for a large integer $N > 0$. Letting $a_1, a_2, \ldots, a_{2KN}$ be the sequence of values, the vector for item i is defined as follows. The first component is $\frac{1}{K}$, and each one of the two other components is equal to a_i. Let γ be a threshold such that if the ith constructed value is small, it holds that $0 < a_i < \frac{\gamma}{10K}$ and otherwise $\gamma < a_i < \varepsilon$. The input up to this point is denoted by I_0.

▶ **Lemma 13.** *The optimal cost for packing I_0 is $2N$.*

Let X denote the number of bins used by the algorithm for the first part of the input and by definition this is also the number of large items. Let $\mu = \frac{\gamma}{10}$. Thus, every K small values have total value below μ.

The second part of the input

For the value of μ that is based on the action of the algorithm, we define the next part of the input. There are N items of each one of the two types: $(0, 1 - 4 \cdot \mu, \mu)$ and $(0, \mu, 1 - 4 \cdot \mu)$. So, in total there are $2N$ items of these types. The input at this time is called I_1, i.e. I_1 is the input consisting of the first two parts of the input together.

Two offline packings of I_1

We define two offline packings, for which the first part of the input is packed in a fixed manner. For each possibility of the third part of the input, we will use one of those offline packings that we present here. Consider the first part of the input (the items of I_0), and separate small items from large items. Large items are packed such that every bin has K of them (where one such bin may have a smaller number of these items). The large items require $\lceil \frac{X}{K} \rceil$ bins. These are feasible bins because none of the components of the sum of any bin is above 1, since no component of any item of the first part is above $\frac{1}{K}$. Small items are also packed K in each bin, and there are at most $2N$ such bins since the total number of items for the first part is $2KN$. The total number of small items is $2KN - X$. The first component of the bin has load 1, but the other components have loads below μ. Every such bin can also receive one item of each type of the second part. In one packing, we partition the items of the second part into pairs where every pair consists of items of different types. In this offline packing each pair is packed together, these pairs are first packed into bins with K small items of the first part, and if there are any unpacked items of the second part, they are packed into new bins (also in pairs). In the second packing, every bin gets just one such item of the second part. For both offline packings, the numbers of bins do not exceed $\frac{X}{K} + 1 + 2N$.

The third part of the input

The third part of the input may contain two alternative sets of items. In the first case, leading to the input I_{21}, there are N items of the type $(0, 1 - \mu, 1 - \mu)$. Every such item is packed into a different bin by any algorithm. In the offline packing, these items are packed first into bins with (at most) K small items of the first part (but without large items of the first part and without any items of the second part) and then into new bins. The online algorithm cannot combine such items with any item into the same bin, as we will see.

In the second case, leading to the input I_{22}, there are N items of each of the types: $(0, 3\mu, 1 - 2\mu)$, $(0, 1 - 2\mu, 3\mu)$. No pair of such items can be packed into one bin, but it can

join a bin with (at most) K small items of the first part and one item of the second part (of the suitable type) but without large items of the first part. It also cannot be packed with an item of the other type of the second part.

This concludes the description of the input construction. Next, we turn our attention to proving the resulting lower bound on the asymptotic competitive ratio for the case $d = 3$.

Proving the resulting lower bound

Here, we prove the following result.

▶ **Theorem 14.** *There is no online algorithm for the case $d = 3$ whose asymptotic competitive ratio is smaller than $\frac{9}{4}$.*

A very slight improvement over the lower bound which we proved above in Theorem 14 can be obtained as follows, similarly to the known construction for $d = 2$ [6]. An alternative second part of the input will contain items whose first component is not zero but some multiple of $\frac{1}{K}$. The second component will be slightly larger than $\frac{1}{3}$, where there will be large items and small items (small items are those that are packed by the algorithm into a bin that cannot receive another item), and all these items have second components larger than $\frac{1}{3}$. There may be a third part of the input (in this alternative input), similarly to the construction of [6]. We omit the details as the idea is similar and the improvement is very small. We note that this value of 2.25 is a valid lower bound for the cases $d \geq 4$ as well. For the case $d = 5$ we could get the same lower bound by another method in Section 3 as well, where we proved significantly larger values for larger dimensions.

5 The case $d = 8$

We consider this special case as well, in order to demonstrate that the asymptotic competitive ratio grows relatively fast with the dimension. We picked the value of $d = 8$ as for this dimension we are able to exhibit new properties of instances leading to improved lower bounds. Once again the lower bound construction consists of three parts.

The first part of the input

The first part of the construction is identical to that of the case $d = 3$, including the property that the values of K and ε are the same, with the only change that the components equal to a_i are not just the second and third components, but all components with indexes $2, 3, \ldots, 7$ are equal to a_i. The first component is still $\frac{1}{K}$, while the 8th component is equal to zero. The values γ and μ are defined as in the first part of the construction for the case $d = 3$.

The second part of the input

The second part of the input consists of $6N$ items consisting of six groups each of which has N vectors, where every N vectors of a common group are identical. All these vectors have 8th components equal to $\frac{1}{3}$ and first components equal to zero. The other components are equal to either 0 or to $1 - 3\mu$, where every item has exactly one component equal to $1 - 3\mu$, and we will call it the large component of the item. We will have the items of group j (for $j = 1, 2, \ldots, 6$) having component $j + 1$ equal to $1 - 3\mu$ while all other components (excluding the 8th component) are zero.

Analyzing the packing of the algorithm at the end of the second part

Before describing the third part of the input, we introduce some notation and properties of the packing of the algorithm at the end of the second part of the input. The items of the second part are defined so that no bin can have more than three such items by the constraint on the 8th component, and all (at most three) items of one bin have distinct large components since $1 - 3\mu > \frac{1}{2}$. We will distinguish the cases where a bin contains three, two, or just one item of the second part of the input, introducing notation for their corresponding bin numbers.

For $j_1, j_2, j_3 \in \{2, 3, 4, 5, 6, 7\}$, where $j_1 < j_2 < j_3$, let X_{j_1,j_2,j_3} be the number of bins with (exactly) three items of the second part of the input, whose large components are j_1, j_2, and j_3. There are 20 such variables. For $j_4, j_5 \in \{2, 3, 4, 5, 6, 7\}$, where $j_4 < j_5$, let Y_{j_4,j_5} be the number of bins with (exactly) two items of the second part of the input, whose large components are j_4 and j_5. There are 15 such variables. For $j_6 \in \{2, 3, 4, 5, 6, 7\}$, let Z_{j_6} be the number of bins with (exactly) one item of the second part of the input, whose large component is j_6. There are six such variables. Since every bin opened by the algorithm for the first part of the input has a sum of components of items above $\gamma = 10\mu$ in components $2, 3, \ldots, 7$, all these bins of the algorithm are new.

The number of bins opened by the algorithm for the first part of the input is denoted by Q and it satisfies

$$Q \geq 2N \ .$$

The sum of variables of the form X_{j_1,j_2,j_3} is denoted by X, the sum of variables of the form Y_{j_4,j_5} is denoted by Y, and the sum of variables of the form Z_{j_6} is denoted by Z. That is, $X = \sum_{j_1,j_2,j_3} X_{j_1,j_2,j_3}$, $Y = \sum_{j_4,j_5} Y_{j_4,j_5}$, and $Z = \sum_{j_6} Z_{j_6}$. By counting the number of items of the second part, we have

$$3X + 2Y + Z = 6N \ .$$

The third part of the input

The third part has one of ten possible sets of items, of similar structures. These items have six non-zero components, which are components $2, 3, \ldots, 7$. Every item has three components whose values are 2μ, and three components whose values are $1 - \mu$. No two such items can be packed into the same bin since the sum of such a component for two items is either $2\mu + 1 - \mu > 1$ or $2(1 - \mu) > 1$. For a triple $\{2, j_7, j_8\}$ where $j_7, j_8 \in \{3, 4, 5, 6, 7\}$ are fixed component indexes, and $j_7 < j_8$, the input consists of N items whose components $2, j_7, j_8$ are equal to $1 - \mu$ and the components $\{2, 3, 4, 5, 6, 7\} \setminus \{2, j_7, j_8\}$ are equal to 2μ, and N items whose components $2, j_7, j_8$ are equal to 2μ and the components $\{2, 3, 4, 5, 6, 7\} \setminus \{2, j_7, j_8\}$ are equal to $1 - \mu$. This completes the construction of the input. Next, we prove the resulting lower bound.

Proving the resulting lower bound

Recall the decision variables X, Y, Z, Q whose values are determined by the algorithm but they satisfies the conditions $Q \geq 2N$ and $3X + 2Y + Z = 6N$ established above. We first upper bound the optimal offline cost after the third part of the input and then present a lower bound on the maximum cost of the algorithm on these 10 inputs that can be constructed in the third part of the input. As in the proof for $d = 3$ we can assume $\frac{Q}{K} \leq 6$.

▶ **Lemma 15.** *For each of the ten inputs that can be created at the end of the third part, there is an offline solution whose cost is at most* $2N + 7$.

The algorithm can combine into a common bin some items of the third part with items of the second part but it cannot use bins that were used for items of the first part for packing items of the second or third parts. We describe only bins without any items of the first part because any bin of the algorithm containing an item of the first part cannot receive any additional items. We discuss such bins with two or three items of the second part (since bins with just one item can always receive additional items). For a set $\{2, j_7, j_8\}$, there may be bins with two items of the second part, where one of the items has a large component in the set $\{2, j_7, j_8\}$ and the other one has a large component in the set $\{2, 3, 4, 5, 6, 7\} \setminus \{2, j_7, j_8\}$. In addition, there may be bins with three items of the second part, where the set of large components is none of the sets $\{2, j_7, j_8\}$ and $\{2, 3, 4, 5, 6, 7\} \setminus \{2, j_7, j_8\}$ (comparing them as sets and not as ordered tuples).

By Lemma 15, for large values of N we find for the asymptotic competitive ratio R that

$$Q + X + Y + Z \leq 2RN \ .$$

We introduce two new variables Y', X' where Y' is the number of bins of the algorithm with two items of the second part that cannot receive an item of the third part, and X' is the number of bins of the algorithm with three items of the second part that cannot receive an item of the third part (for the choice of third part, that is, we fix the third part temporarily). Then, by considering this input using the fact that the third part of the input requires packing items into at least $2N$ bins and we cannot use $Q + Y' + X'$ of the bins which were opened for the first or second parts, we conclude that

$$Q + Y' + X' + 2N \leq 2RN \ .$$

We take the sum of the last inequality for all ten options of j_7, j_8, where the right hand side is $20RN$, and the multiplier of N on the left hand side is $20N$. Since we consider all options for the third part of the input, the values X' and Y' can have different values, and more precisely, each one has up to ten different values.

We count the multiplier of each variable as follows. Variables of the form X_{j_1,j_2,j_3} are included in all variables X' except for the option where j_1, j_2, j_3 are components of equal values, (the algorithm chose exactly the same subset as the one chosen for the third part of the input) which is just one case of the third part. Thus, the multiplier of X_{j_1,j_2,j_3} is 9. Variables of the form Y_{j_4,j_5} are included in all variables Y' except for cases where j_4 and j_5 are components of equal values, which is the case if $\{j_4, j_5\} \subseteq \{2, j_7, j_8\}$ or $\{j_4, j_5\} \subseteq \{2, 3, 4, 5, 6, 7\} \setminus \{2, j_7, j_8\}$. Out of the 15 variables, there are nine such options that are included (in the sense that the bins cannot be used for items of the third part of the input) and six that are not included. Thus, every variable is included in six of the ten partitions, and in this sum of constraints every variable Y_{j_4,j_5} has a multiplier of 6. We get

$$10Q + 6Y + 9X + 20N \leq 20RN \ . \tag{1}$$

Using $3X + 2Y + Z = 6N$ and $Q + X + Y + Z \leq 2RN$, we find by subtraction that $2X + Y - Q \geq 6N - 2RN$ or alternatively

$$9X + 4.5Y - 4.5Q \geq 27N - 9RN \ .$$

By subtracting the last inequality from (1), we have $1.5Y + 14.5Q + 47N \leq 29RN$. Since $Y \geq 0$ and $Q \geq 2N$ hold, we establish that $76N \leq 29RN$ and therefore $R \geq \frac{76}{29} \approx 2.620689655$ as we summarize in the following theorem.

▶ **Theorem 16.** *There is no online algorithm for the case $d = 8$ whose asymptotic competitive ratio is smaller than $\frac{76}{29} \approx 2.620689655$.*

References

1. Y. Azar, I. R. Cohen, A. Fiat, and A. Roytman. Packing small vectors. In *Proc. of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA'16)*, pages 1511–1525, 2016.
2. Y. Azar, I. R. Cohen, S. Kamara, and F. B. Shepherd. Tight bounds for online vector bin packing. In *Proc. of the 45th ACM Symposium on Theory of Computing (STOC'13)*, pages 961–970, 2013.
3. Y. Azar, I. R. Cohen, and A. Roytman. Online lower bounds via duality. In *Proc. of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA'17)*, pages 1038–1050, 2017.
4. L. Babel, B. Chen, H. Kellerer, and V. Kotov. Algorithms for on-line bin-packing problems with cardinality constraints. *Discrete Applied Mathematics*, 143(1-3):238–251, 2004.
5. J. Balogh, J. Békési, Gy. Dósa, L. Epstein, and A. Levin. A new and improved algorithm for online bin packing. In *Proc. of the 26th European Symposium on Algorithms (ESA'18)*, pages 5:1–5:14, 2018.
6. J. Balogh, J. Békési, Gy. Dósa, L. Epstein, and A. Levin. Online bin packing with cardinality constraints resolved. *Journal of Computer and System Sciences*, 112:34–49, 2020.
7. J. Balogh, J. Békési, Gy. Dósa, L. Epstein, and A. Levin. A new lower bound for classic online bin packing. *Algorithmica*, 83(7):2047–2062, 2021.
8. J. Balogh, J. Békési, Gy. Dósa, J. Sgall, and R. van Stee. The optimal absolute ratio for online bin packing. *Journal of Computer and System Sciences*, 102:1–17, 2019.
9. J. Balogh, L. Epstein, and A. Levin. Truly asymptotic lower bounds for online vector bin packing. *CoRR*, abs/2008.00811, 2020. `arXiv:2008.00811`.
10. N. Bansal, A. Caprara, and M. Sviridenko. A new approximation method for set covering problems, with applications to multidimensional bin packing. *SIAM Journal on Computing*, 39(4):1256–1278, 2009.
11. N. Bansal and I. R. Cohen. An asymptotic lower bound for online vector bin packing. *CoRR*, abs/2007.15709, 2020. `arXiv:2007.15709`.
12. N. Bansal, M. Eliás, and A. Khan. Improved approximation for vector bin packing. In *Proc. of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA'16)*, pages 1561–1579, 2016.
13. D. Blitz. Lower bounds on the asymptotic worst-case ratios of on-line bin packing algorithms. Technical Report 114682, University of Rotterdam, 1996. M.Sc. thesis.
14. D. Blitz, A. van Vliet, and G. J. Woeginger. Lower bounds on the asymptotic worst-case ratio of online bin packing algorithms. Unpublished manuscript, 1996.
15. A. Caprara, H. Kellerer, and U. Pferschy. Approximation schemes for ordered vector packing problems. *Naval Research Logistics*, 92:58–69, 2003.
16. C. Chekuri and S. Khanna. On multidimensional packing problems. *SIAM Journal on Computing*, 33(4):837–851, 2004.
17. L. Epstein. Online bin packing with cardinality constraints. *SIAM Journal on Discrete Mathematics*, 20(4):1015–1030, 2006.
18. L. Epstein and A. Levin. AFPTAS results for common variants of bin packing: A new method for handling the small items. *SIAM Journal on Optimization*, 20(6):3121–3145, 2010.
19. W. Fernandez de la Vega and G. S. Lueker. Bin packing can be solved within $1 + \varepsilon$ in linear time. *Combinatorica*, 1(4):349–355, 1981.
20. G. Galambos, H. Kellerer, and G. J. Woeginger. A lower bound for online vector packing algorithms. *Acta Cybernetica*, 10:23–34, 1994.
21. M. R. Garey, R. L. Graham, and D. S. Johnson. Resource constrained scheduling as generalized bin packing. *Journal of Combinatorial Theory Series A*, 21(3):257–298, 1976.

22 M. M. Halldórsson and M. Szegedy. Lower bounds for on-line graph coloring. *Theoretical Computer Science*, 130(1):163–174, 1994.

23 D. S. Johnson. Fast algorithms for bin packing. *Journal of Computer and System Sciences*, 8:272–314, 1974.

24 D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey, and R. L. Graham. Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM Journal on Computing*, 3:256–278, 1974.

25 N. Karmarkar and R. M. Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (FOCS'82)*, pages 312–320, 1982.

26 L. T. Kou and G. Markowsky. Multidimensional bin packing algorithms. *IBM Journal of Research and Development*, 21(5):443–448, 1977. doi:10.1147/rd.215.0443.

27 L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete mathematics*, 13(4):383–390, 1975.

28 S. Sandeep. Almost optimal inapproximability of multidimensional packing problems. *CoRR*, abs/2101.02854, 2021. arXiv:2101.02854.

A Adaptive constructions for deterministic algorithms

In addition to the lower bound for a general d, we present lower bounds that achieve better guarantees for relatively small values of d versus deterministic algorithms. The lower bounds are based on a method that produces a sequence of values with several important properties, and are produced by an adversary according to the algorithm's behavior. Specifically, we define inputs using a method presented in the past [6]. In this method, a binary condition on the assignment of every item is defined (e.g., whether the item is assigned into a new bin), and it is used by the adversary (who presents the input) for the definition of the properties of the following item. More precisely, the adversary keeps an active interval of (scalar) values (contained in $(0, 1)$), and it modifies the interval after the assignment of every input item by the algorithm. These values are not necessarily the actual sizes of the input items, even in the one-dimensional case, although item sizes are based on them in a simple pre-specified way. This means that the generated value is not necessarily the size of the new input item, nor will it always be equal to a component of its vector. Nevertheless, this generated value is used in the definition of the new item, for example, it may be subtracted from some fixed value, or added to a fixed size.

The number of required items is decided in advance, or (in some cases) an upper bound on the number of required items is provided in advance in cases where the exact number of required items is revealed later on. This number is used to decide upon the initial interval of the values, as well. The initial interval is also based on the required sizes and properties of the values. The initial interval is always defined such that the smallest size is strictly positive and the largest size is sufficiently small.

Input items are presented one by one. After the assignment of an item by the algorithm, the validity of the condition is tested for that item. During the process of input construction, it is ensured (via a process resembling binary search, or geometric binary search) that values corresponding to items satisfying the binary condition are larger by a pre-specified (constant) multiplicative factor than the value of any item not satisfying the binary condition. In this way, the process determines two regions, as explained below. We will call the resulting ranges of values *large* and *small*, where the two ranges are disjointed. Items with large values, i.e., from the large range, are called large, and items with small values, i.e., from the small range, are called small.

Note that when an item is presented, its size is defined without any prior knowledge of the assignment, so it is still unknown at that moment whether the binary condition holds for this item. Thus, its value is defined without any knowledge of its dimensions. That knowledge is gained based on the action of the algorithm once the item is packed. Based on the packing, if its value is required to be large, future values will be much smaller, and if its value is required to be small, future values will be much larger.

The construction allows us to define positive values smaller than a given value $\varepsilon > 0$, such that for a pre-defined (constant) multiplicative factor k, any large value is more than k times larger than any small value. Thus, there is a value $\gamma < \varepsilon$ such that every small value is smaller than $\frac{\gamma}{k}$ and every large value is larger than γ. If items are one dimensional, and their sizes are simply these values, it would imply that, for example, an item of size $1 - \gamma$ can be packed with k small items, but cannot be packed with one large item into the same bin. Note that in this case large items are also quite small, although not as small as the small items. It is possible to define items differently in one dimension, i.e., not only in the way that their sizes are equal to the values. One option is to use the values as complements of sizes (to 1). Another option is to use an additive term, for example, items can have sizes of $\frac{1}{3}$ plus the defined value. In this case, one can define a value of ε such that $\frac{1}{3} + \varepsilon < \frac{1}{2}$, for example. For items that are vectors, one can define a part of the components to be determined based on the corresponding value. For example, it is possible in the case $d = 5$ that two components will be equal to the value, while three other components are equal to zero.