Randomized Algorithms for Online Vector Load Balancing

Yossi Azar[*]
Tel-Aviv University

Ilan Reuven Cohen[†]
The Hebrew University of Jerusalem

Debmalya Panigrahi[‡]
Duke University

Abstract

We study randomized algorithms for the online vector bin packing and vector scheduling problems. For vector bin packing, we achieve a competitive ratio of $\tilde{O}(d^{1/B})$, where d is the number of dimensions and B the size of a bin. This improves the previous bound of $\tilde{O}(d^{1/(B-1)})$ by a polynomial factor, and is tight up to logarithmic factors. For vector scheduling, we show a lower bound of $\Omega(\frac{\log d}{\log \log d})$ on the competitive ratio of randomized algorithms, which is the first result for randomized algorithms and is asymptotically tight. Finally, we analyze the widely used "power of two choices' algorithm for vector scheduling, and show that its competitive ratio is $O(\log \log n + \frac{\log d}{\log \log d})$, which is optimal up to the additive $O(\log \log n)$ term that also appears in the scalar version of this algorithm.

1 Introduction

Online load balancing problems, where a sequence of jobs arriving online must be irrevocably assigned to a set of machines/bins, has been a vibrant area of research for many decades. There are two classical variants of this problem. If the size of each bin is fixed and the goal is to minimize the number of bins used, then the problem is called *online bin packing* and has been studied since the work of Johnson in the 70s (see, e.g., [26, 27] and surveys [21, 34] for later work). On the other hand, if the number of machines is fixed, and the goal is to minimize the maximum load on any machine, then the problem is called *list scheduling* (or *online makespan minimization*) and has been studied since the work of Graham in the 60s (see, e.g., [23, 24] and surveys [3, 33, 2] for later work). Much of the recent interest in these problems has been for the *vector bin packing* (VBP) and *vector scheduling* (VS) problems, where jobs have vector loads and the goal is to balance the load on all dimensions simultaneously [6, 5, 28, 25]. This is motivated by applications such as data center scheduling, where each job requires multiple resources (each resource is a dimension) for its execution.

In this paper, we give new randomized online algorithms and lower bounds for the VBP and VS problems. For the VBP problem, we close the polynomial gap between the best upper and lower bounds known previously, by giving a new randomized algorithm whose competitive ratio[1] matches the lower bound (up to a log factor). For the VS problem, we give the first lower bound for randomized algorithms, showing that randomization cannot improve the competitive ratio vis-à-vis the best known deterministic bounds. Nevertheless, in practice, simple randomized allocations are often preferred, and the "power of two choices" paradigm has been a popular choice in (scalar) scheduling algorithms. We give the first analysis of this algorithm in the context of vector scheduling, and show that as in the scalar case, it gives an exponential advantage over a naïve random assignment. We describe these problems and our results below.

Online Vector Bin Packing (VBP). *A sequence of job vectors, each of d dimensions, have to be assigned online to bins of size B in each dimension. (By scaling, we assume that each job vector is in $[0,1]^d$.) The goal is to minimize the number of bins used.*

For the VBP problem, the simple greedy algorithm has a competitive ratio of $O(d)$ [22]. In the offline setting, this was improved to a $O(1 + \epsilon d + \log(1/\epsilon))$-approximation in [16]. On the other hand, [6], using ideas from [17], showed that a $d^{1-\epsilon}$-approximation would imply $NP = ZPP$. For fixed d (i.e., super-polynomial in d running times), the approximation factor improves to $\ln d + O(1)$ [9, 8, 16]. To overcome the strong lower bound for arbitrary d, Epstein [18] initiated the study of VBP with variable bin sizes. In the online setting, the $O(d)$ bound was improved to $\tilde{O}(d^{\frac{1}{B-1}})$ for $B \geq 2$ in [6], who also showed an information-theoretic lower bound of $\Omega\left(d^{\frac{1}{B}-\epsilon}\right)$ (for any $\epsilon > 0$). Note the polynomial gap between these two bounds, which we close in this paper. For large enough B, the lower bound vanishes; for this case, [5] gave a $(1+\epsilon) \cdot e$-competitive algorithm if $B = (1/\epsilon^2) \cdot \Omega(\log d)$ (for any $\epsilon > 0$). If

[*]email:azar@tau.ac.il. Supported in part by the Israel Science Foundation (grant No. 1506/16) and by the ICRC Blavatnik Fund.

[†]email:ilanrcohen@gmail.com. Supported in part by the I-CORE program (center no. 4/11).

[‡]email:debmalya@cs.duke.edu. Supported in part by NSF grants CCF-1527084 and CCF-1535972.

[1]The *competitive ratio* for a minimization problem is the maximum, over all instances, of the ratio between the objective of the online algorithm and that of an optimal (offline) solution (see, e.g., the book by Borodin and El-Yaniv [15]).

vectors are splittable, [5] gave an e-competitive algorithm, and a matching lower bound was shown in [7].

Our Results. For the (online) VBP problem, we give a simple randomized algorithm that obtains a competitive ratio of $\tilde{O}(d^{1/B})$ for any bin size B, which matches (up to a log factor) the existing lower bound of $\Omega(d^{1/B})$ [6]. This closes the polynomial gap between this lower bound and the best upper bound of $\tilde{O}(d^{1/(B-1)})$ known earlier [6]. In particular, for $B = 2$, the best algorithm known previously was the naïve greedy algorithm with a competitive ratio of $O(d)$ (which applies to any value of B); our algorithm improves the competitive ratio to $\tilde{O}(\sqrt{d})$. For binary job vectors (i.e., in $\{0,1\}^d$), the competitive ratio of our algorithm improves to $\tilde{O}(d^{1/(B+1)})$. This is also tight (up to logarithmic factors) due to a matching lower bound of $\Omega(d^{1/(B+1)})$ for binary vectors [6].

THEOREM 1.1. (SECTION 2) *There is a randomized algorithm for the* VBP *problem with a competitive ratio of* $O(d^{1/B}\log d)$. *For binary job vectors, the competitive ratio improves to* $O(d^{1/(B+1)}\log d)$.

Online Vector Scheduling (VS). *A sequence of job vectors, each of d dimensions, have to be assigned online to a fixed set of machines.The goal is to minimize the* makespan *of the assignment, i.e., the maximum load across all dimensions and machines.*

Chekuri and Khanna [16] introduced the offline VS problem, and obtained an $O(\log^2 d)$-approximation algorithm and a PTAS for fixed d, whose running time was later improved in [10]. Various generalizations of the offline VS problem have also been studied [11, 19, 20, 14]). For the online setting, random assignment yields a competitive ratio of $O(\log dm)$ [16, 38], which was improved to $O(\log d)$ by [6] and [28], and finally to $O\left(\frac{\log d}{\log\log d}\right)$ by [25], the last paper also showing that this bound is asymptotically tight for deterministic algorithms. Prior to our work, no lower bound was known for randomized algorithms.

Our Results. For the (online) VS problem, we give the first lower bound for randomized algorithms. Our randomized lower bound matches the known deterministic (upper and lower) bound of $\Theta(\frac{\log d}{\log\log d})$ [25].

THEOREM 1.2. (SECTION 3) *There is a (information-theoretic) lower bound of* $\Omega(\frac{\log d}{\log\log d})$ *on the competitive ratio of randomized algorithms for the* VS *problem.*

While the above result shows that randomization cannot improve the competitive ratio for the VS problem, simple randomized allocations are often preferred in practice, particularly in large systems without central control. In particular, the "power of two choices" (POTC) paradigm [4, 30], where for each job, two machines are chosen independently at random and the job is assigned to the one with smaller load, has been widely used in scalar scheduling (and many other contexts), In this paper, we give the first analysis of this paradigm for scheduling vectors, where two machines are chosen independently at random and the job is assigned to the one with smaller maximum load across all dimensions. We show that this algorithm has a competitive ratio of $O(\log\log n + \frac{\log d}{\log\log d})$, where n is the number of machines. This analysis is tight: the first term matches the well-known and tight scalar bound of $O(\log\log n)$ [4], while the second term matches our randomized lower bound for the VS problem. Moreover, just like in the scalar case, this gives an exponential improvement over a random assignment, whose competitive ratio is $\Omega(\frac{\log n}{\log\log n} + \frac{\log d}{\log\log d})$. To the best of our knowledge, this is the first analysis of the POTC paradigm for assigning vectors instead of scalars. Given the vast literature on POTC for scalar assignment and its many variants (e.g., [29, 1, 36, 12, 13, 35, 32]), we hope that this work will initiate further study in the role of POTC for vector assignments.

THEOREM 1.3. (SECTION 4) *The competitive ratio of the "power of two choices" algorithm for the* VS *problem is* $O(\log\log n + \frac{\log d}{\log\log d})$.

Our Techniques. We now outline the main techniques used to obtain the above results.

Vector Bin Packing. First, we describe the algorithm for the VBP problem (Theorem 1.1). In this outline, we will assume that the value of the optimal solution (OPT) is known to the algorithm; in the actual algorithm, we only use the maximum volume in a dimension to lower bound OPT, and this can be estimated using a standard guess-and-double trick. First, consider a random assignment of the vectors to $\beta \cdot$ OPT bins. Let p_β denote the probability that any vector *overflows*, i.e., cannot fit in its randomly chosen bin. Our goal is to choose β large enough so that OPT for the overflow vectors is at most a constant fraction of the original OPT. If we can ensure this, then we can simply recurse on the overflow vectors.

So, let us try to bound the value of p_β. Observe that the probability of overflow of a vector on a fixed dimension is given by an approximate Gaussian tail. However, the width of the tail varies for different vectors, and also across different dimensions of the same vector. To overcome this difficulty, we perform a convex decomposition of each vector into a restricted set of vectors, each of which has the same value on all its non-zero dimensions. We show that the probability p_β of the original job overflowing is bounded by a convex combination of the corresponding probabilities for the restricted vectors. This allows us to focus only the restricted instance. However, the different vectors still have different non-zero values, which correspond to different Gaussian tail probabilities.

Next, we round up all dimensions of all vectors to powers of e^{-r}, for $r = 0, 1, \ldots$. In general, such a rounding can affect the value of OPT arbitrarily, but since we only use maximum total volume on a dimension to lower bound OPT, this surrogate is only affected by a constant. This reduces the number of different job sizes, and allows us to apply the tail bound on each dimension for individual job sizes separately. To obtain the overall probability of overflow, we must now take a union bound across all dimensions. Instead of taking the union bound directly, we separate jobs into large and small ones based on total volume. The large jobs are the ones that have a sufficient volume for us to reserve a space of 1^d in a bin – for these jobs, we use a simple $\mathrm{FirstFit}$ strategy instead of the randomized algorithm. For the small jobs, we now obtain a tighter union bound; e.g., for binary job vectors, if small jobs have at most a volume of d', then the union bound needs to be taken over d' dimensions only. For smaller vectors, the benefit is not as straightforward since the smaller volume might be distributed across the d dimensions, but this makes each individual dimension smaller thereby allowing a tighter tail probability. Overall, we show that the overflow probability p_β is bounded away from 1 for $\beta = C \cdot d^{1/B}$, for a large enough constant C.

Unfortunately, ensuring that each vector fits in its randomly chosen bin with constant probability is not sufficient. Consider the extreme case where each job has a unit load on only a single dimension. Let X_k denote the random variable representing the number of jobs with dimension k in the recursive instance. Clearly, X_k is stochastically dominated by a Binomial random variable $\mathtt{Bin}(B \cdot \mathrm{OPT}, p_\beta)$, whose expected value is $p_\beta \cdot B \cdot \mathrm{OPT}$. But, OPT for the recursive instance is given by $\max_k(X_k/B)$, whose expectation can be higher than $\max_k \mathbb{E}[X_k/B] = p_\beta \cdot \mathrm{OPT}$ and is not necessarily a constant fraction of OPT. To overcome this difficulty, observe that the probability of $\mathtt{Bin}(B \cdot \mathrm{OPT}, p_\beta)$ exceeding $C' \cdot B \cdot \mathrm{OPT} \cdot p_\beta$ for a constant C' is (approximately) equal to the Gaussian tail on $\gamma := B \cdot \mathrm{OPT} \cdot p_\beta$. If $\gamma = \Omega(\log d)$, then the tail probability is $1/\mathrm{poly}(d)$, which allows us to use the union bound across the d dimensions. More generally, we repeat the initial random assignment $\alpha := \log d/\mathrm{OPT}$ times, and a vector overflows to the next recursive level only if it fails to get assigned in any of the random trials. This ensures that OPT in the recursive instance is only a constant fraction of OPT in the original instance. The detailed algorithm and analysis appear in Section 2.

Vector Scheduling. Let us now turn to the VS problem. Our first goal is to obtain a lower bound for randomized algorithms (Theorem 1.2). We use Yao's minimax principle [37] which allows us to show a corresponding lower bound for deterministic algorithms given a random input distribution. Our lower bound construction is for the *monochromatic clique* (MC) problem, where vertices of a graph arrive online and must be colored from a fixed set of t colors so that the size of the largest monochromatic clique is minimized. It turns out that a $\Omega(t^c)$ lower bound for any constant $c > 0$ for the MC problem translates to an $\Omega(\frac{\log d}{\log\log d})$ lower bound for the VS problem (under certain conditions) [25]; so we focus on this problem instead (for other connections between online lower bounds for coloring and vector load balancing, see [16, 6]).

Unfortunately, the deterministic lower bound from [25] does not generalize to the randomized setting. Let us denote the color classes used by the algorithm as *bins*. The lower bound graph in [25] was constructed in a manner that the vertices in a bin form a set of disjoint cliques, and any new vertex adds to one of the cliques of the bin it is placed in. Clearly, this requires that the adjacencies of the new vertex depend on the bins chosen by the algorithm for the previous vertices. Eventually, one of these cliques in some bin grows to $\sqrt{t}$ vertices, and the main technical work was in showing that by the time this happens, an optimal coloring OPT would only have created constant-sized monochromatic cliques. For the randomized lower bound, however, the instance must be oblivious to the algorithm. Hence, we cannot deterministically guarantee that the new vertex will always add to an existing clique; instead, with appropriate modifications, we can guarantee this only with probability $1/t$. With this weaker guarantee, we now need to increase the number of vertices by a factor of t, but this creates a large monochromatic clique in OPT as well.

Our new construction takes a completely different approach (indeed, it gives an alternative, simpler deterministic lower bound construction as well). Observe that the randomized construction can hide very little information from the deterministic algorithm: in particular, we assume that the algorithm precisely knows the optimal coloring for all previous vertices when it attempts to color the new vertex. In fact, in our construction, the new vertex reveals a pair of colors to the deterministic algorithm and promises that a randomly chosen color among the two will be used to color the vertex in OPT. (Note that this implies for every vertex, the deterministic algorithm can match the color of OPT with probability $1/2$!) Moreover, the adjacencies are very simple: every vertex v_i is connected to all previous vertices $v_1, v_2, \ldots, v_{i-1}$ for which OPT uses any color *other than* the two candidate colors for v_i. This completes the construction of the graph.

Note that the adjacency rule ensures that OPT is a proper coloring, i.e., the optimal size of the largest monochromatic clique is 1. For the algorithm, note that its monochromatic cliques correspond to cliques on vertices in the same bin. To lower bound the size of the largest such clique, we design a meta-graph for each bin whose meta-vertices correspond to the colors used by OPT on the vertices in the bin, and meta-edges correspond to the candidate color-pairs for these vertices. By controlling the total number of vertices and the choice of candidate color-pairs, we show that the meta-graph in some bin has an induced subgraph that implies a monochromatic clique of size $\Omega(\sqrt{t})$ in the algorithm's solution.

Finally, consider the "power of two choices" (POTC) algorithm for the VS problem (Theorem 1.3). The algorithm is simple: for each job, choose two bins uniformly at random and assign to the one where the maximum load across all dimensions (called *height* of the vector) is smaller. In this outline, let us restrict ourselves to binary vectors for simplicity, although the same ideas (with some additional complications) work for general vectors. First, let us first remind ourselves of the idea behind the classic proof of an $O(\log \log n)$ bound for the single-dimensional case (also called balls in bins) [4]. Suppose p_i is the probability that a ball has height at least i. Then, the fraction of bins with height i is also at most p_i. As a result, the probability that a ball has height $i+1$ or higher is given by $p_{i+1} = p_i^2$ since it chooses two bins of height i (or higher). Because of this quadratic relationship, the probability decays at a double exponential rate with increasing height and vanishes at height $O(\log \log n)$. But now, we have binary vectors instead of balls, and this quadratic relationship no longer holds in general. To see this, note that we might now have as many as nd vectors in n bins. Thus, if p_i is the probability that a vector has height i or higher, then the fraction of bins with height at least i can be $d \cdot p_i$, which gives $p_{i+1} = d^2 \cdot p_i^2$. Intuitively, the analysis fails to distinguish between sparse vectors and dense ones. In effect, it treats all binary vectors as the all-ones vector, potentially increasing the optimal value by a factor of d.

For a more refined analysis, let us first categorize the vectors into *low* and *high* ones, based on whether their height is smaller or greater than $C \cdot \frac{\log d}{\log \log d}$ for some constant C. What is the probability of a job being high? The above analysis does not give a satisfactory answer. However, for a uniform random assignment, we can apply Chernoff bounds since the allocations are independent. To relate these two processes, we show that the height distribution produced by the POTC algorithm is stochastically dominated by that produced by a random assignment (after constant scaling). This implies that the probability of a job being high is bounded by $1/\text{poly}(d)$. In other words, the above strategy of treating all vectors as the all-ones vector is usable for the high vectors, but not for the low vectors. We use this fact to recover the doubly exponential rate of decay on the probabilities *beyond the height of* $C \cdot \frac{\log d}{\log \log d}$. Further details appear in Section 4.

2 Vector Bin Packing

Let $\mathbb{V} = \{v_1, v_2, \ldots\}$ be the set of d dimensional vectors ($v_i \in [0,1]^d$) that arrive online, and $BINS = \{BIN_1, BIN_2, \ldots\}$ be the set of d dimensional bins with capacity B in each dimension that these vectors need to be packed into. Our goal is to minimize the number of bins used for this packing, i.e., the cardinality of the set $BINS$.

We denote the total volume on dimension k of the vectors $\mathbb{V}$ by $V_k(\mathbb{V}) := \sum_i v_{i,k}$, and use Λ to denote the maximum volume across all dimensions, i.e., $\Lambda(\mathbb{V}) := \max_k\{V_k\}$. (We omit $\mathbb{V}$ from the notations if the set of vectors is clear from the context.) For an allocation of the vectors to the bins, we denote the load of bin BIN_j on dimension k by $L_{j,k} := \sum_{i: v_i \in BIN_j} v_{i,k}$. Let $\text{OPT}(\mathbb{V})$ be the number of bins used by an optimal (offline) assignment for $\mathbb{V}$. Note that for every dimension k, $\text{OPT}(|V|) \geq V_k(\mathbb{V})/B$ by volume conservation:

OBSERVATION 1. *For any set of vectors* $\mathbb{V}$, *we have* $\text{OPT}(\mathbb{V}) \geq \Lambda(\mathbb{V})/B$.

We distinguish between vectors that have a large volume and those that do not: we say that a vector v_i is *large* if $\sum_k v_{i,k} \geq H$ for a value H that will be defined later; if a vector is not large, then it is called *small*. To understand the intuition, think of a bin as being divided into B slots, each of unit volume in all dimensions. Since all vectors are in $[0,1]^d$, a naïve greedy packing of vectors would, in the worst case, pack a single vector into each such slot. This is not desirable for small vectors, since most of the volume in a slot is then wasted, but is sufficient for large vectors. Hence, large vectors can be packed using any simple greedy strategy. We use a distinct set of bins (denoted L and S respectively) to allocate large and small vectors. For the large vectors, we use a FirstFit algorithm that allocates the next large vector to the first existing bin in L that can accommodate it, and opens a new bin if none can. On the other hand, for the small vectors, we use a randomized allocation algorithm called SmallVectorFit described later that requires most of our work.

The next lemma formalizes the above intuition about large vectors.

LEMMA 2.1. *The number of bins used by* FirstFit *for allocating large vectors is at most* $(d/H) \cdot \text{OPT}(\mathbb{V}) + 1$.

Proof. First, observe that the number of large vectors in $\mathbb{V}$ is at most $(d/H) \cdot B \cdot \text{OPT}(\mathbb{V})$. This follows from volume conservation: since $\text{OPT}(\mathbb{V})$ bins can accommodate all the vectors in $\mathbb{V}$, their total volume $\sum_k V_k$ is at most $d \cdot B \cdot \text{OPT}(\mathbb{V})$, which is divided by the minimum volume H of large vectors to give the above bound. Next, we note that in L, every bin has at least B vectors, except possibly the last one. This follows from the fact that *any* B vectors fit in a bin; hence, when a new bin is opened, all the previous bins must contain at least B vectors for them not to be able to accommodate the new vector. The number of bins in L is then bounded by the ratio of the number of large vectors and the number of vectors per bin, i.e., $(d/H) \cdot \text{OPT}(\mathbb{V}) + 1$.

The threshold H is chosen as $d^{\frac{B-1}{B}}$ and $d^{\frac{B}{B+1}}$ for general and binary vectors respectively. Note that this yields the desired competitive ratios of $d^{\frac{1}{B}}$ and $d^{\frac{1}{B+1}}$ in the above lemma for large vectors. Our goal in the rest of this section is to design an algorithm $\mathrm{SmallVectorFit}$ for small vectors that opens $O((d/H)\cdot(\Lambda/B))$ bins in expectation, and thereby yields the desired competitive ratio for small vectors by Observation 1. In describing the algorithm, we will assume that it knows the value of Λ. This can be simulated with constant factor loss using a standard guess-and-double trick: every time Λ doubles, the algorithm stops assigning vectors to the existing bins, and a new set of bins corresponding to the new value of Λ is introduced.

First, we analyze a uniform random assignment of small vectors to $T(\Lambda) := C\cdot(d/H)\cdot(\Lambda/B)$ bins. We give the analysis for binary vectors followed by the analysis for general vectors.

Uniform Random Assignment for Binary Small Vectors.

Let $T_1(\Lambda) = \frac{C_1\cdot e\cdot\Lambda\cdot d^{\frac{1}{B+1}}}{B}$ for some constant C_1; this represents the number of bins in the u.a.r. assignment for binary vectors. First, we analyze the probability of a bin overflowing in a given dimension k.

LEMMA 2.2. *Let a set of binary vectors $\mathbb{V}$ with $\Lambda = \Lambda(\mathbb{V})$ be assigned uniformly at random to $T_1(\Lambda)$ bins. Then, for any dimension k, we have $\mathbb{P}[L_{j,k}\geq B] \leq \frac{1}{C_1\cdot d^{\frac{B}{B+1}}}$.*

Proof. By linearity of expectation,

$$\mu := \mathbb{E}[L_{j,k}] = \frac{\sum_i v_{i,k}}{T_1(\Lambda)} \leq \frac{\Lambda}{T_1(\Lambda)} = \frac{B}{C_1\cdot e\cdot d^{\frac{1}{B+1}}}.$$

Applying Chernoff bounds (see, e.g., [31]), with $(1+\delta)\cdot\mu = B$, i.e., $1+\delta = C_1\cdot e\cdot d^{\frac{1}{B+1}}$, we have

$$\mathbb{P}[L_{j,k}\geq B] = \mathbb{P}[L_{j,k}\geq(1+\delta)\cdot\mu] < \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{\mu} \leq \left(\frac{e}{1+\delta}\right)^{(1+\delta)\cdot\mu} = \left(\frac{e}{1+\delta}\right)^{B} \leq \frac{1}{C_1\cdot d^{\frac{B}{B+1}}}.$$

The overflow probability of small vectors can now be bounded as a simple corollary of the above lemma.

LEMMA 2.3. *The probability that a small binary vector $v_i\in V$ (i.e., a binary vector with at most $d^{\frac{B}{B+1}}$ ones) overflows in a uniform random assignment to $T_1(\Lambda)$ bins is at most $1/C_1$.*

Proof. Let BIN_j be chosen by the u.a.r. assignment for vector v_i. Then, by union bound,

$$\mathbb{P}[v_i \text{ overflows}] \leq \frac{1}{T_1(\Lambda)}\cdot\sum_{j=1}^{T_1(\Lambda)}\sum_{k:v_{i,k}=1}\mathbb{P}[L_{j,k}\geq B] \leq d^{\frac{B}{B+1}}\cdot\frac{1}{C_1\cdot d^{\frac{B}{B+1}}} = \frac{1}{C_1}.$$

Uniform Random Assignment for General Small Vectors. Let $T_2(\Lambda) = \frac{C_2\cdot e\cdot\Lambda\cdot d^{\frac{1}{B}}}{B}$ for some constant C_2; this represents the number of bins in the u.a.r. assignment for general vectors. Our first goal is to show that the probability of a vector not fitting in its prescribed bin, i.e., of *overflowing* in the random assignment, is bounded away from 1. We distinguish between vectors based on the size of the individual non-zero dimensions. We round up all non-zero values in the vectors to the nearest value of the form e^{-r}, for an integer r. Recall that we use the volume bound $\Lambda(\mathbb{V})/B$ to lower bound $\mathrm{OPT}(\mathbb{V})$, and this lower bound only increases by a factor of e due to this rounding. Hence, in the rest of the analysis, we will assume that we are given a rounded instance.

Now, for a vector of size e^{-r} in a dimension k (we call this a category-r dimension for the vector), a bin is said to be *full* if its load in dimension k exceeds $B-e^{-r}$. Our first goal is to bound the probability of the load of a bin exceeding $B-e^{-r}$ in a fixed dimension. We will denote this bound by $\beta_r := \frac{2}{C_2\cdot d^{(B-e^{-r})/B}}$.

LEMMA 2.4. *Let a set of vectors $\mathbb{V}$ in $[0,1]^d$ with $\Lambda=\Lambda(\mathbb{V})$ be assigned uniformly at random to $T_2(\Lambda)$ bins. Then, for any dimension k and for any integer r, we have $\mathbb{P}[L_{j,k}\geq B-e^{-r}]\leq\beta_r$.*

Proof. By linearity of expectation, $\mu := \mathbb{E}[L_{j,k}] = \frac{\sum_i v_{i,k}}{T_2(\Lambda)} \leq \frac{\Lambda}{T_2(\Lambda)} = \frac{B}{C_2\cdot e\cdot d^{\frac{1}{B}}}$. Applying Chernoff bounds (see, e.g., [31]), with $(1+\delta_r)\cdot\mu = B-e^{-r}$, i.e., $1+\delta_r = C_2\cdot e\cdot d^{\frac{1}{B}}\cdot(1-e^{-r}/B)$. Thus,

$$\frac{e}{1+\delta_r} = \frac{1}{C_2\cdot d^{\frac{1}{B}}\cdot(1-e^{-r}/B)} \leq \frac{2}{C_2\cdot d^{\frac{1}{B}}} \leq \beta_r^{\frac{1}{B-e^{-r}}}.$$

Then, we have

$$\mathbb{P}[L_{j,k}\geq B-e^{-r}] = \mathbb{P}[L_{j,k}\geq(1+\delta_r)\cdot\mu] < \left(\frac{e^{\delta_r}}{(1+\delta_r)^{(1+\delta_r)}}\right)^{\mu} \leq \left(\frac{e}{1+\delta_r}\right)^{(1+\delta_r)\cdot\mu} = \left(\frac{e}{1+\delta_r}\right)^{B-e^{-r}} \leq \beta_r.$$

We will now use the above lemma to bound the overflow probability of small vectors. Before giving the formal

Algorithm 1 The recursive $\mathrm{SmallVectorFit}_\Lambda$ algorithm on the input $\mathbb{V}$ with $\Lambda(\mathbb{V}) \le \Lambda$.

```
• Initialize:
    – $\alpha \leftarrow \lceil \frac{2e \ln d}{\Lambda} + 2 \rceil$
    – $BINS := \{BINS_r : 1 \le r \le \alpha\}$, where $BINS_r := \{BIN_{r,1}, BIN_{r,2}, \ldots, BIN_{r,T(\Lambda)}\}$
    – Overflow $\leftarrow \emptyset$
• Assignment of vector $v_i \in \mathbb{V}$:
    – Choose $j$ u.a.r. in $\{1, 2, \ldots, T(\Lambda)\}$ for $r \in \{1, 2, \ldots, \alpha\}$
    – Add $v_i$ to any bin $BIN_{r,j_r}$; if it fits in none, add $v_i$ to Overflow and assign $v_i$ by $\mathrm{SmallVectorFit}_{\Lambda/2}$
    – if for any dimension $k$, $V_k(Overflow) > \Lambda/2$
          then Assign the rest of the vectors via $\mathrm{FirstFit}$
```

proof, we outline the main ingredients. First, we apply Lemma 2.4 to each dimension of the vector individually. Now, we need to apply a union bound and sum over the tail probabilities from the individual dimensions – this turns out to be the main challenge. We split the total volume of the vector into the contributions from the category-r dimensions for different values of r. These volumes give a corresponding bound on the number of dimensions in each category, which allows us to use the union bound separately on each category. However, for very small sizes, this bound turns out to be weak. To overcome this problem, we merge all the categories representing very small sizes into a single category representing a moderate size. Next, we show that for all categories that have at least moderate size, the union bound gives a strong bound if only a single category is present in the vector. Finally, we use convexity to extend this result to an arbitrary mixture of categories, thereby proving the lemma.

LEMMA 2.5. *The probability that a small vector $v_i \in V$ (i.e., a vector with at most $d^{\frac{B-1}{B}}$ in total volume) overflows in a uniform random assignment to $T_2(\Lambda)$ bins is at most $\frac{4e}{C_2}$.*

Proof. We partition the dimensions based on the value of $v_{i,k}$. Let $C(r) := \{k : v_{i,k} = e^{-r}\}$ denote category-r dimensions, i.e., dimensions with value e^{-r} (recall that all values are of the form e^{-r} in the rounded instance that we are operating on). Let $v_i(r)$ denote the total volume of vector i in category-r dimensions, i.e., $v_i(r) := \sum_{k \in C(r)} v_{i,k}$, and let the number of such dimensions be denoted $Q_r := |C(r)|$. Note that $Q_r \cdot e^{-r} = v_{i,r}$. It will also be useful to separately consider dimensions with very small values in v_i; for this purpose, let us define a threshold $\tilde{r} := \lfloor \frac{\ln d}{B} \rfloor$. Category-$r$ dimensions for $r > \tilde{r}$ will be considered separately in this analysis.

The probability that the vector v_i overflows is now given by:

$$\mathbb{P}[v_i \text{ overflows}] \le$$
$$\frac{1}{T_2(\Lambda)} \cdot \sum_{j=1}^{T_2(\Lambda)} \sum_{r \ge 0} \sum_{k \in C(r)} \mathbb{P}[L_{j,k} \ge B - e^{-r}] =$$
$$\frac{1}{T_2(\Lambda)} \cdot \sum_{j=1}^{T_2(\Lambda)} \left(\sum_{r=0}^{\tilde{r}} \sum_{k \in C(r)} \mathbb{P}[L_{j,k} \ge B - e^{-r}] + \right.$$
$$\left. \sum_{r > \tilde{r}} \sum_{k \in C(r)} \mathbb{P}[L_{j,k} \ge B - e^{-r}] \right).$$

Using Lemma 2.4 and the fact that $\beta_r < \beta_{\tilde{r}}$ for $r > \tilde{r}$, we get

$$\mathbb{P}[v_i \text{ overflows}] \le \sum_{r=0}^{\tilde{r}} Q_r \cdot \beta_r + d \cdot \beta_{\tilde{r}} =$$
$$d^{\frac{B-1}{B}} \left(\sum_{r=0}^{\tilde{r}} \alpha_r \cdot e^r \cdot \beta_r + d^{1/B} \cdot \beta_{\tilde{r}} \right),$$

where $\alpha_r := Q_r \cdot e^{-r} / d^{\frac{B-1}{B}}$.

Note that, $\alpha_r = v_{i,r}/d^{\frac{B-1}{B}}$, which implies $\sum_r \alpha_r \le 1$ for small vectors. This allows us to view the first term as a convex combination of $e^r \cdot \beta_r$. For the second term, note that $d^{1/B} < e^{\tilde{r}}$. Hence,

$$\mathbb{P}[v_i \text{ overflows}] \le 2 \cdot \max_{0 \le r \le \tilde{r}} \left(d^{\frac{B-1}{B}} \cdot e^r \cdot \beta_r \right).$$

So, we are left to bound the value of $d^{\frac{B-1}{B}} \cdot e^r \cdot \beta_r$ for $0 \le r \le \tilde{r}$. Note that $d^{\frac{B-1}{B}} \cdot e^r \cdot \beta_r = (2/C_2) \cdot \gamma_r$, where $\gamma_r := d^{\frac{e^{-r}-1}{B}} \cdot e^r$; hence, we proceed to bound the value of γ_r. Let us rewrite $\gamma_r = \frac{a^{x_r}}{a x_r}$, where $a := d^{1/B}$ and $x_r := e^{-r}$. We are interested in the range $x_r \in [1/a, 1]$.

We take the second derivative of γ_r w.r.t. x_r:

$$\frac{d^2\gamma_r}{dx_r^2} = \frac{a^{x_r-1}(x_r^2\ln^2 a - 2x_r\ln a + 2)}{x_r^3} = \frac{a^{x_r-1}((x_r\ln a - 1)^2 + 1)}{x_r^3} > 0.$$

Hence, γ is a convex function of x_r. This allows us to only consider the two ends of the range: $\gamma_r = 1$ for $x_r = 1$ and $\gamma_r = a^{1/a} \leq e^{1/e} < e$ for $x_r = 1/a$. Thus, $\gamma_r \leq e$ in the range $x_r \in [1/a, 1]$, i.e., $d^{\frac{B-1}{B}} \cdot e^r \cdot \beta_r \leq 2e/C_2$ for $0 \leq r \leq \tilde{r}$. Therefore,

$$\mathbb{P}[v_i \text{ overflows}] \leq 2 \cdot \max_{r\in[0,\tilde{r}]} \left(d^{\frac{B-1}{B}} \cdot e^r \cdot \beta_r\right) \leq 4e/C_2.$$

Allocation Algorithm for Small Vectors (SmallVectorFit**).** Now, we use the observations made above for u.a.r. assignments to design an algorithm (we call this the SmallVectorFit algorithm) for allocating the small vectors. Our algorithm is recursive. Let $\mathbb{V}$ be the input vectors to the algorithm and let $\Lambda := \Lambda(\mathbb{V})$. Instead of assigning the vectors to a single set of $T(\Lambda)$ bins (recall that $T(\Lambda) = T_1(\Lambda)$ for binary vectors and $T(\Lambda) = T_2(\Lambda)$ for general vectors), the algorithm opens α disjoint sets of $T(\Lambda)$ bins and assigns each vector to a bin chosen u.a.r. and independently in each set. The vector is assigned to any of its chosen bins that can accommodate it. If none of the bins can accommodate the vector, we say that the vector *overflows*. Let $\mathbb{V}'$ denote the set of vectors that overflowed. If $\Lambda(V') \leq \Lambda(\mathbb{V})/2$, then we simply run the same algorithm recursively on $\mathbb{V}'$. Note that in this case, the number of bins allocated by the algorithm in the next recursive level decreases by a factor of 2; hence, the total number of bins used across all recursive levels is at most twice that used in the top layer. On the other hand, if $\Lambda(\mathbb{V}') \geq \Lambda(\mathbb{V})/2$, then we subsequent vectors using the $\mathrm{FirstFit}$ algorithm on a new set of bins. (Recall that the $\mathrm{FirstFit}$ algorithm assigns the current vector to the first existing bin that can accommodate it, and opens a new bin if none of the existing bins can fit the vector.) The SmallVectorFit algorithm is given in Algorithm 1. Although we described it as an offline algorithm, note that it can be easily run online with all the recursive levels operating simultaneously.

First, we bound the expected number of bins used by $\mathrm{FirstFit}$, which may be invoked by SmallVectorFit.

LEMMA 2.6. *The expected number of bins opened by the* $\mathrm{FirstFit}$ *algorithm is at most* $2(\Lambda/B) + 1$.

Proof. First, we show that if invoked, $\mathrm{FirstFit}$ opens at most $2d \cdot (\Lambda/B) + 1$. If a bin has load less than $B - 1$ on every dimension, then it can accept *any* vector. Hence, when the $\mathrm{FirstFit}$ starts a new bin, every existing bin must have a total load of at least $B - 1$. The total volume of all vectors is at most $d \cdot \Lambda$. Therefore, $\mathrm{FirstFit}$ opens at most $d \cdot (\Lambda/(B-1)) + 1 \leq 2d \cdot (\Lambda/B) + 1$ bins.

Next, we show that the probability that SmallVectorFit invokes $\mathrm{FirstFit}$ is at most $1/d$. By setting the constants C_1 and C_2 appropriately in Lemmas 2.3 and 2.5 respectively, we bound the probability of v_i not fitting its chosen bin in a single uniform assignment to $1/e$. Therefore, the probability that v_i does not fit in any of the α bins chosen for it is at most $e^{-\alpha}$. This implies that for a given dimension k, the expected volume of vectors in the set Overflow if given by $\mu \leq e^{-\alpha} \cdot \Lambda$. We now use Chernoff bounds with $(1+\delta) := \Lambda/(e\mu) \geq e^{\alpha-1}$. Then, we have

$$\mathbb{P}[V_k(\text{Overflow}) \geq \Lambda/e] \leq \left(\frac{e}{1+\delta}\right)^{(1+\delta)\mu} \leq \left(\frac{e}{e^{\alpha-1}}\right)^{\Lambda/e} \leq e^{\frac{-(\alpha-2)\Lambda}{e}} \leq \frac{1}{d^2},$$

using the value of α. The bound on the probability of invoking $\mathrm{FirstFit}$ now follows by using the union bound over d dimensions.

The lemma follows from the above two observations.

Finally, we bound the expected total number of bins opened by the SmallVectorFit algorithm. Recall that $H = d^{\frac{B}{B+1}}$ for binary vectors and $H = d^{\frac{B-1}{B}}$ for general vectors, and our desired competitive ratio is $O(d/H)$.

LEMMA 2.7. *The expected number of bins opened by the* $\mathrm{SmallVectorFit}$ *algorithm is* $O(d/H) \cdot \frac{\Lambda(\mathbb{V})}{B} + \zeta$ *bins in expectation, where* $\mathbb{V}$ *is the input set of vectors and the additive term* $\zeta = O(d/H) \cdot \frac{\ln d}{B} \cdot O(\log\log d)$.

Proof. We already bounded the the expected number of bins opened by $\mathrm{FirstFit}$ in the above lemma. The number of bins opened in the random assignment in a single recursive level

$$\alpha \cdot T(\Lambda) \leq \left\lceil \frac{2e\ln d}{\Lambda} + 2 \right\rceil \cdot O(d/H) \cdot \frac{\Lambda}{B} = O\left(d/H\right) \cdot (1/B) \cdot \max\{\ln d, \Lambda\}.$$

Note that Λ decreases by a constant factor in each recursive level; hence, the total number of bins opened as long as $\Lambda > \ln d$ is given by $O\left(d/H\right) \cdot \frac{\Lambda(\mathbb{V})}{B}$, where $\mathbb{V}$ is the input set of vectors. There are only $O(\log\log d)$ recursive levels after Λ has dropped to less than $\ln d$, and each such layer opens $O\left(d/H\right) \cdot \frac{\ln d}{B}$ bins for a total of $O\left(d/H\right) \cdot \frac{\ln d}{B} \cdot O(\log\log d)$ bins. This completes the proof of the lemma.

Note that if $\Lambda < \log d$, then the number of recursive levels in the SmallVectorFit algorithm is actually $\ln\Lambda \leq \Lambda$, and therefore, the additive term $\zeta \leq O(d\log d/H) \cdot \frac{\Lambda}{B}$. On the other hand, if $\Lambda \geq \log d$, then $\zeta = O(d/H) \cdot (1/B) \cdot O(\log\log d) \leq O(d\log d/H) \cdot \frac{\Lambda}{B}$. Hence, in either case, the SmallVectorFit algorithm has a competitive ratio of $O(d\log d/H)$ (Theorem 1.1).

3 Lower bound for Vector Scheduling

In this section, we present a tight lower bound for randomized algorithms for the VS problem. We will first show the lower bound for the online monochromatic clique problem, and relate it later to the VS problem.

Minimum Monochromatic Clique (MC). In this problem, the goal is to color the vertices of a given graph using a fixed set of t colors so that the size of the maximum monochromatic clique is minimized.

We consider the online version of this problem, where a new vertex is presented in every online step and the algorithm must color it using one of the t colors $T = \{1, \ldots, t\}$. Our goal is to show a lower bound of $\Omega(\sqrt{t})$ for this problem. Using Yao's minimax principle, we are interested in showing a lower bound for the expected competitive ratio of a deterministic algorithm for a given probability distribution of inputs graphs. We interpret the lower bound construction as a game between Alice (the adversary) and Bob (the algorithm). Alice produces the random graph instance and an optimal coloring OPT for it, while Bob produces a coloring of the graph. To distinguish between the two colorings, we say that Bob places vertices in t bins, each corresponding to a color, while Alice colors vertices to produce OPT. The game has the following repeated steps: (a) Alice present a vertex v_i and its edges to previous vertices $\{v_1, v_2, \ldots, v_{i-1}\}$, (b) Bob chooses one of the t bins for vertex v_i, and (c) Alice reveals the color of vertex v_i in OPT. Importantly, steps (a) and (c) performed by Alice must be oblivious to step (b) for the current or any previous vertex. On the other hand, in step (b), we assume that Bob knows everything in the past, i.e., the color of all previous vertices in OPT.

In fact, in our construction, we restrict Alice further: she reveals two possible colors for the current vertex in step (a) and commits to using one of them uniformly at random in step (c). For vertex v_i, these two colors are denoted by the unordered pair (c_i^1, c_i^2), where $c_i^1, c_i^2 \in T$. For step (a), Alice uses the following rule: *the new vertex v_i is connected to all previous vertices that did not get colors c_i^1, c_i^2 OPT*. Since Alice will use one of these two colors for v_i in OPT, the following observation is immediate.

OBSERVATION 2. *There does not exist any monochromatic edge in OPT, i.e., OPT is a proper coloring.*

Next, Bob places vertex v_i in some bin in step (b), and as promised earlier, Alice colors v_i in OPT with one of c_i^1 or c_i^2 with probability $1/2$ each (and independent of the color of any other vertex) – we denote the color of v_i in OPT by $\text{OPT}(v_i)$.

In order to complete the description, we need to specify how Alice chooses the two colors c_i^1 and c_i^2. This is done deterministically. We order all the pairs among the t colors arbitrarily, and call the sequence $\mathbb{P} = \{P_1, P_2, \ldots\}$, where $|\mathbb{P}| = \binom{t}{2}$. Alice presents $\binom{t}{2} \cdot (t+1)$ vertices in total, grouped into $\binom{t}{2}$ phases where phase r comprises $t+1$ vertices with color-pair P_r. We denote $v_i \sim P_r$ if vertex v_i belongs to phase r.

We call a color-pair *good* for a bin B if there exists two vertices $v_i, v_j \sim P_r$ such that $B(v_i) = B(v_j) = B$ and $\text{OPT}(v_i) \neq \text{OPT}(v_j)$. The next lemma claims that across all bins, the number of good pairs is large whp. [2]

LEMMA 3.1. *The number of good pairs across all bins is at least $t(t-1)/8$ whp.*

Proof. Using the pigeonhole principle, for any color-pair P_r, there exists two vertices $v_i, v_j \sim P_r$ such that $B(v_i) = B(v_j)$. Furthermore, since OPT colors vertices randomly among the two candidate colors, $\text{OPT}(v_i) \neq \text{OPT}(v_j)$ with probability at least $1/2$. By linearity of expectation, the expected number of good pairs is $\mu \geq \frac{\binom{t}{2}}{2} = t(t-1)/4$. Now, we apply Chernoff bounds with $(1+\delta)\cdot\mu := t(t-1)/8$ to obtain the lemma.

LEMMA 3.2. *If there are k good pairs in a bin, then the bin contains a clique of size at least $\sqrt{k/2}$.*

To prove this lemma, the following simple graph theoretic property will be useful.

FACT 3.1. *In any graph with m edges, at least one of the following must hold: (a) there is a matching of size $\sqrt{m/2}$, or (b) there is a vertex with degree at least $\sqrt{m/2}$.*

Proof. If the degree of every vertex is less than $\sqrt{m/2}$, then form a matching of size $\sqrt{m/2}$ by repeatedly picking an arbitrary edge and discarding all its adjacent edges in the graph.

Using this fact, we prove Lemma 3.2.

Proof. [Proof of Lemma 3.2] Fix a bin B. Create a meta-graph $H = (C, F)$, where F is the set of good color-pairs in bin B and C are the colors in these pairs. Note that the number of edges in this meta-graph is $|F| = k$. Now, apply Fact 3.1 on H.

Suppose there exists a matching M of $\sqrt{k/2}$ good color-pairs in H. Let (c_1, c_1') and (c_2, c_2') be two different color-pairs in M. Then, there exist vertices v_i, v_j in bin B such that $\text{OPT}(v_i) = c_1$ and $\text{OPT}(v_2) = c_2$, and neither corresponds to the color-pair (c_1, c_2). This implies that there is an edge (v_1, v_2) in the input graph G. Since M has $\sqrt{k}$ edges, there is a clique of size $\sqrt{k/2}$ among vertices in bin B in the input graph G.

Next, suppose there is a color $c \in C$ such that it has at least $\sqrt{k/2}$ neighbors in H. Let us denote these neighbors $c_1, c_2, \cdots, c_\ell$, where $\ell \geq \sqrt{k/2}$. Now, consider the vertices

[2]whp = with high probability, i.e. with probability $1 - o(1)$

$v_1, v_2, \ldots, v_\ell$, where $v_i \sim (c, c_i)$ and $\textsc{opt}(v_i) = c_i$. Clearly, these vertices form an $\ell(\geq \sqrt{k/2})$-sized clique in the input graph G.

Let k_B be the random variable denoting the number of good color-pairs in bin B, and let $k := \max_B k_B$. From Lemma 3.1, $\sum_B k_B \geq t(t-1)/8$ whp. Thus, $k = \max_B k_B \geq (1/t) \cdot \sum_B k_B \geq (t-1)/8$. By Lemma 3.2, this implies that the size of the largest clique in any bin is at least $\sqrt{\frac{t-1}{16}}$ whp.

Reduction from MC **to** VS **lower bound.** We will now give a reduction from an instance of the MC problem to an instance of the VS problem, so that our MC lower bound translates to a lower bound for VS, and yields Theorem 1.2. This reduction is similar to previous reductions from online coloring to the VS problem [16, 25].

Proof. [Proof of Theorem 1.2] Recall that every vertex is labeled by a pair of colors deterministically, of which one is chosen at random to color in the optimal solution. Denote all $\sqrt{\frac{t-1}{16}}$-sized subsets of vertices by $\mathbb{A} = \{A_1, A_2, \ldots\}$. Since there are $(t+1) \cdot \binom{t}{2} = O(t^3)$ vertices, we have $|\mathbb{A}| \leq t^{O(\sqrt{t})}$. Our VS instance uses t machines, each corresponding to a color, and $d = |\mathbb{A}| = t^{O(\sqrt{t})}$ dimensions, each corresponding to a $\sqrt{\frac{t-1}{16}}$-sized set of vertices $A \in \mathbb{A}$.

Vectors vec_i in the VS problem correspond to vertices v_i in the MC problem. All the vectors are binary vectors. For a vertex v_i, the corresponding vector vec_i has a 1 in dimension A if two conditions are satisfied: (a) $v_i \in A$, and (b) all vertices in A that appeared before v_i (including v_i), i.e., the set $|A \cap \{v_1, v_2, \ldots, v_i\}|$ forms a clique in the graph. If any of these conditions is violated, then $\text{vec}_{i,\langle r,A\rangle} = 0$.

The above properties imply that a monochromatic clique of size $\sqrt{\frac{t-1}{16}}$ in some color class c corresponds to a load of $\sqrt{\frac{t-1}{16}}$ on the dimension representing the subset of vertices in the clique, and vice-versa. We have already established that there is a monochromatic clique of size $\sqrt{\frac{t-1}{16}}$ in the deterministic algorithm for MC whp. This implies a load of $\sqrt{\frac{t-1}{16}}$ on some dimension in the deterministic algorithm for VS whp. This establishes a lower bound of $\Omega(\sqrt{t})$ on the expected competitive ratio of any randomized algorithm for the VS problem. To complete the proof, note that $d = t^{O(\sqrt{t})} = \sqrt{t}^{O(\sqrt{t})}$, which implies that the lower bound for the VS problem is $\Omega\left(\frac{\log d}{\log\log d}\right)$.

4 Vector Scheduling: Power of Two Choices Algorithm

In this section, we consider the "power of two choices" (POTC) algorithm for the VS problem. The input comprises a sequence of m vectors $\mathbb{V}$, where each vector $v_i \in [0,1]^d$. The set of bins is denoted $BINS = \{BIN_1, BIN_2, \ldots, BIN_n\}$, i.e., n denotes the number of bins. For each vector, the algorithm chooses two bins BIN_{j_1} and BIN_{j_2} uniformly at random, and assigns the vector to the one among the two $j \in \{j_1, j_2\}$ that minimizes $\max_k L_{j,k}$ after the assignment (ties are broken arbitrarily). Our goal in this section is to prove Theorem 1.3, which gives a bound of $O(\log\log n + \frac{\log d}{\log\log d})$ on the competitive ratio of the POTC algorithm.

By scaling all vectors by a factor of OPT, we assume w.l.o.g that $\textsc{opt} \leq 1$. This implies that for all dimensions k, we have $\sum_i v_{i,k} \leq n$. Let us define the k-height of a vector v_i assigned to BIN_j as the value of $L_{j,k}$ immediately before assigning vector v_i to it. Let $\tau := 6e \cdot \frac{\log d}{\log\log d}$; our first goal is to bound the probability of a vector having k-height greater than τ for any dimension k.

LEMMA 4.1. *For any vector $v_i \in \mathbb{V}$, the probability that its k-height exceeds τ for some dimension k is at most* $1/(e \cdot d^2)$.

Proof. Fix a dimension k. We compare the POTC assignment for $\mathbb{V}$ to an alternate random process for a set of vectors $\mathbb{V}^{(2)}$, which contains two copies of each vector in $\mathbb{V}$. Every time a vector $v_i \in \mathbb{V}$ chooses two bins, say BIN_{j_1} and BIN_{j_2}, in POTC, we assign the two copies of v_i in $\mathbb{V}^{(2)}$ to bins BIN_{j_1} and BIN_{j_2} in the alternate random process. Clearly, the k-height of v_i in POTC is at most the maximum height of its two copies in $\mathbb{V}^{(2)}$. But, observe that the vectors in $\mathbb{V}^{(2)}$ are actually being assigned uniformly at random. Hence, for any height h and dimension k, the number of vectors with k-height greater than h in the POTC assignment for $\mathbb{V}$ is stochastically dominated by the number of vectors with k-height greater than h in a uniform random assignment of $\mathbb{V}^{(2)}$ on the same set of bins. We now apply Chernoff bounds with expectation $\mu \leq 2$ and δ chosen such that $(1+\delta)\cdot\mu = \tau$ (note that $1 + \delta \geq \tau/2$) to the u.a.r. process:

$$\begin{aligned}(4.1)\qquad \mathbb{P}[L_{j,k} \geq \tau] &\leq \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu \leq \\ \left(\frac{e}{1+\delta}\right)^{(1+\delta)\cdot\mu} &\leq (2e/\tau)^\tau \leq 1/(e \cdot d^3),\end{aligned}$$

using the value of τ. The lemma now follows by using a union bound across the d dimensions.

We say that a vector is *high* if its k-height exceeds τ for some dimension k; else, the vector is said to be *low*. The set of high and low vectors are respectively denoted H and L. We need to bound the load due to high vectors. Let H_j denote the high vectors assigned to BIN_j, and $L^h_{j,k} := \sum_{i:v_i \in H_j} v_{i,k}$ denote the load in BIN_j on dimension k due to high vectors. Note that our goal is to bound $\max_k L^h_{j,k} = \max_k \left(\sum_{i:v_i \in H_j} v_{i,k}\right)$. Instead, we bound

$L_j^* = \sum_{i:v_i \in H_j} \max_k (v_{i,k}) = \sum_{i:v_i \in H_j} v_i$ where $v_i = \max_k (v_{i,k})$ (also note that $\sum_i v_i \le \sum_{i,k} v_{i,k} \le dn$.) In other words, we think of high vectors as single-dimensional vectors of size equal to their maximum dimension. This is clearly an overestimate; nevertheless, we show that even this slack bound on the maximum load due to high vectors is $O(\log \log n)$ whp.

We define the *surplus height* of a high vector assigned to BIN_j as the value of L_j^* immediately before the allocation. Define $\beta_0 = 1/(e \cdot d^2)$ and $\beta_r := e \cdot d \cdot \beta_{r-1}^2 = (1/e) \cdot d^{-2^r - 1}$. Note that $\beta_r < 1/n$ for $r = \log \log n + O(1)$. We will show that β_r bounds the probability of the surplus height of a vector exceeding $2r$. The proof of the next lemma is similar to [4].

LEMMA 4.2. *For any $r \ge 0$, the total volume of vectors with surplus height $2r$ is at most $n \cdot \beta_r$ whp. In particular, there is no vector of k-height greater than $\tau + 2 \log \log n + O(1)$ for any dimension k, whp.*

Proof. We use induction on r. The base case, i.e., $r = 0$, follows from Lemma 4.1.

Let h_i be the surplus height of vector v_i. Let $\nu_{\ge 2r}(i-1)$ be the number of bins of surplus height at least $2r$ when vector i is being assigned. Since the two choices for a vector are independent, we have:

$$\mathbb{P}(h_i \ge 2r \mid \nu_{\ge 2r}(i-1)) = \frac{\left(\nu_{\ge 2r}(i-1)\right)^2}{n^2}.$$

Let $\mathcal{E}_r$ be the event that $\nu_{\ge 2r}(m) \le n \cdot \beta_r$, i.e., the inequality holds for the last vector (recall that m is the total number of vectors). Clearly, $\mathcal{E}_r$ implies that $\nu_{\ge 2r}(i-1) \le n \cdot \beta_r$ for any i. Recall that Λ is the maximum volume on any dimension (w.l.o.g., $\Lambda \ge n$ since we can add dummy vectors otherwise to satisfy this condition). Now fix $r \ge 0$ and consider a series of binary random variables Y_i^r for $i = 1, 2, \ldots, m$, where

$$Y_i^r = 1 \textbf{ iff } h_i \ge 2r \textbf{ and } \nu_{\ge 2r}(i-1) \le n \cdot \beta_r,$$
$$Y_i^r = 0, \text{ otherwise}.$$

In other words, Y_i^r is 1 if and only if the surplus height of vector v_i is at least $2r$ despite the number of bins with surplus height $2r$ or greater being less than $n \cdot \beta_r$.

Let ω_i represent the choices available to the vector v_i. Clearly

$$\mathbb{P}(Y_i^r = 1 \mid \omega_1, \ldots, \omega_{i-1}) \le \beta_r^2. \tag{4.2}$$

Consider the random variable $\sum_i |v_i| \cdot X_i^r$, where $|v_i| = \max_k v_{ik}$ and X_i^r are i.i.d variables with $X_i^r = 1$ with probability β_r^2 and 0 otherwise. Clearly, for every possible history up to (but not including) i the conditional probability that $Y_i^r = 1$ is at most the conditional probability that $X_i^r = 1$ (both are $\{0, 1\}$ variables). Hence $\sum_i |v_i| \cdot X_i^r$ stochastically dominates $\sum_i |v_i| \cdot Y_i^r$, and hence we conclude that

$$\mathbb{P}\left(\sum |v_i| \cdot Y_i^r \ge k\right) \le \mathbb{P}\left(\sum |v_i| \cdot X_i^r \ge k\right) \forall k. \tag{4.3}$$

Let $\mu_{\ge 2r}$ be the total volume of vectors with surplus height at least $2r$. Then, conditioned on $\mathcal{E}_r$, we have that $\mu_{\ge 2r} \le \sum_i |v_i| \cdot Y_i^r$. Therefore:

$$\mathbb{P}(\mu_{\ge 2r} \ge k \mid \mathcal{E}_r) \le \mathbb{P}(\sum_i |v_i| \cdot Y_i^r \ge k \mid \mathcal{E}_r) \le \frac{\mathbb{P}(\sum |v_i| \cdot Y_i^r \ge k)}{\mathbb{P}(\mathcal{E}_r)}. \tag{4.4}$$

In addition, the total volume of vectors at surplus height at least $2r$ is an upper bound on the number of bins at surplus height $2r + 2$. Hence,

$$\mathbb{P}(\nu_{\ge 2r+2} \ge k \mid \mathcal{E}_r) \le \mathbb{P}(\mu_{\ge 2r} \ge k \mid \mathcal{E}_r) \tag{4.5}$$

Combining Eqns. (4.3), (4.4), and (4.5), we obtain that

$$\mathbb{P}(\nu_{\ge 2r+2} \ge k \mid \mathcal{E}_r) \le \frac{\mathbb{P}(\sum_i |v_i| \cdot X_i^r \ge k)}{\mathbb{P}(\mathcal{E}_r)}. \tag{4.6}$$

Recall that $\sum_i |v_i| \le dn$ and hence $\mathbb{E}(\sum_i |v_i| \cdot X_i^r) \le \beta_r^2 \cdot dn$. For all i, clearly $|v_i| \le 1$, and we can use Chernoff bounds with $\delta = e - 1$, i.e., set $k = e \cdot \beta_r^2 \cdot dn$, to get

$$\mathbb{P}(\sum_i |v_i| \cdot X_i^r \ge e \cdot \beta_r^2 \cdot dn) \le e^{-\beta_r^2 \cdot dn}. \tag{4.7}$$

With our choice of the β_r values, we have $e \cdot \beta_r^2 \cdot dn = n \cdot \beta_{r+1}$. Hence, Eq. (4.7) becomes

$$\mathbb{P}\left(\sum |v_i| \cdot X_i^r \ge n \cdot \beta_{r+1}\right) \le e^{-\beta_r^2 \cdot dn}. \tag{4.8}$$

We discard the probability space where $\mathcal{E}_{\ge 0} = \{\nu_0 \le n/(e \cdot d^2)\}$ does not hold (note that this probability space is $o(1)$ by Lemma 4.1), and assume in the rest of the proof that this condition holds with certainty. From Eqn. (4.6) and (4.8), for $r \ge 1$, we have

$$\mathbb{P}(\neg \mathcal{E}_{r+1} \mid \mathcal{E}_r) \le \frac{1}{n^2 \cdot \mathbb{P}(\mathcal{E}_r)}, \text{ provided } \beta_r^2 \cdot dn \ge 2 \ln n.$$

By Bayes rule, we have $\mathbb{P}(\neg \mathcal{E}_{r+1}) \le \mathbb{P}(\neg \mathcal{E}_{r+1} \mid \mathcal{E}_r) \cdot \mathbb{P}(\mathcal{E}_r) + \mathbb{P}(\neg \mathcal{E}_r)$. It follows that if $\beta_r^2 \cdot dn \ge 2 \ln n$, then:

$$\mathbb{P}(\neg \mathcal{E}_{i+1}) \le \frac{1}{n^2} + \mathbb{P}(\neg \mathcal{E}_i), \tag{4.9}$$

which implies that all the events $\mathcal{E}_i$ hold whp.

To complete the proof, we need to consider the case $\beta_r^2 \cdot dn < 2 \ln n$; let r^* be the smallest value of r for which this is true. This implies that $r^* \leq \log\log n + O(1)$. As before,

$$\mathbb{P}(\mu_{\geq 2r^*} \geq 6\ln n \mid \mathcal{E}_{r^*}) \leq \frac{\mathbb{P}(\sum |v_i| \cdot X_i^{r^*} \geq 6\ln n)}{\mathbb{P}(\mathcal{E}_{r^*})} \leq \frac{1}{n^2 \cdot \mathbb{P}(\mathcal{E}_{r^*})},$$

which implies that

$$\mathbb{P}(\nu_{\geq 2r^*+2} \geq 6\ln n) \leq \frac{1}{n^2} + \mathbb{P}(\neg\mathcal{E}_{r^*}).$$

Finally, by the Markov bound,

$$\mathbb{P}(\mu_{\geq 2r^*+2} \geq 1 \mid \ \nu_{\geq 2r^*+2} \leq 6\ln n) \leq \frac{\mathbb{P}(\sum |v_i| \cdot X_i^{r^*+1} \geq 1)}{\mathbb{P}(\nu_{\geq 2r^*+2} \leq 6\ln n)} \leq \frac{dn \cdot (6\ln n/(dn))^2}{\mathbb{P}(\nu_{\geq 2r^*+2} \leq 6\ln n)}$$

which implies that

$$\mathbb{P}(\mu_{\geq 2r^*+2} \geq 1) \leq \frac{(6\ln n)^2}{dn} + \mathbb{P}(\nu_{\geq 2r^*+2} \geq 6\ln n).$$

Combining Eqns. (4.9), (4), and (4.10), we obtain that

$$\mathbb{P}(\nu_{\geq 2r^*+4} \geq 1) \leq \mathbb{P}(\mu_{\geq 2r^*+2} \geq 1) \leq \frac{(6\ln n)^2}{dn} + \frac{r^*+2}{n^2} = o(1),$$

which implies that the maximum surplus height is at most $2r^* + 4 = \log\log n + O(1)$ whp.

This completes the proof of Theorem 1.3.

References

[1] Micah Adler, Soumen Chakrabarti, Michael Mitzenmacher, and Lars Eilstrup Rasmussen. Parallel randomized load balancing. *Random Struct. Algorithms*, 13(2):159–188, 1998.

[2] Susanne Albers. *Introduction to scheduling*. Chapman and Hall/CRC, 2010.

[3] Yossi Azar. On-line load balancing. In *Online Algorithms, The State of the Art*, pages 178–195, 1996.

[4] Yossi Azar, Andrei Z. Broder, Anna R. Karlin, and Eli Upfal. Balanced allocations. *SIAM J. Comput.*, 29(1):180–200, 1999.

[5] Yossi Azar, Ilan Reuven Cohen, Amos Fiat, and Alan Roytman. Packing small vectors. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1511–1525, 2016.

[6] Yossi Azar, Ilan Reuven Cohen, Seny Kamara, and F. Bruce Shepherd. Tight bounds for online vector bin packing. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 961–970, 2013.

[7] Yossi Azar, Ilan Reuven Cohen, and Alan Roytman. Online lower bounds via duality. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain*, pages 1038–1050, 2017.

[8] Nikhil Bansal, Alberto Caprara, and Maxim Sviridenko. A new approximation method for set covering problems, with applications to multidimensional bin packing. *SIAM J. Comput.*, 39(4):1256–1278, 2009.

[9] Nikhil Bansal, Marek Eliás, and Arindam Khan. Improved approximation for vector bin packing. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1561–1579, 2016.

[10] Nikhil Bansal, Tim Oosterwijk, Tjark Vredeveld, and Ruben van der Zwaan. Approximating vector scheduling: Almost matching upper and lower bounds. *Algorithmica*, 76(4):1077–1096, 2016.

[11] Sanjoy K. Baruah, Vincenzo Bonifaci, Gianlorenzo D'Angelo, Alberto Marchetti-Spaccamela, Suzanne van der Ster, and Leen Stougie. Mixed-criticality scheduling of sporadic task systems. In *Algorithms - ESA 2011 - 19th Annual European Symposium, Saarbrücken, Germany, September 5-9, 2011. Proceedings*, pages 555–566, 2011.

[12] Petra Berenbrink, Artur Czumaj, Angelika Steger, and Berthold Vöcking. Balanced allocations: the heavily loaded case. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 745–754, 2000.

[13] Petra Berenbrink, Tom Friedetzky, Zengjian Hu, and Russell A. Martin. On weighted balls-into-bins games. *Theor. Comput. Sci.*, 409(3):511–520, 2008.

[14] Vincenzo Bonifaci and Andreas Wiese. Scheduling unrelated machines of few different types. *CoRR*, abs/1205.0974, 2012.

[15] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[16] Chandra Chekuri and Sanjeev Khanna. On multidimensional packing problems. *SIAM J. Comput.*, 33(4):837–851, 2004.

[17] Brian C. Dean, Michel X. Goemans, and Jan Vondrák. Approximating the stochastic knapsack problem: The benefit of adaptivity. *Math. Oper. Res.*, 33(4):945–964, 2008.

[18] Leah Epstein. On variable sized vector packing. *Acta Cybern.*, 16(1):47–56, 2003.

[19] Leah Epstein and Tamir Tassa. Vector assignment problems: a general framework. *J. Algorithms*, 48(2):360–384, 2003.

[20] Leah Epstein and Tamir Tassa. Vector assignment schemes for asymmetric settings. *Acta Inf.*, 42(6-7):501–514, 2006.

[21] Gabor Galambos and Gerhard J. Woeginger. On-line bin packing – a restricted survey. *Zeitschrift für Operations Research*, 42(1):25–45, 1995.

[22] M. R. Garey, Ronald L. Graham, David S. Johnson, and Andrew Chi-Chih Yao. Resource constrained scheduling as generalized bin packing. *J. Comb. Theory, Ser. A*, 21(3):257–298, 1976.

[23] R. L. Graham. Bounds for certain multiprocessing anomalies. *Siam Journal on Applied Mathematics*, 1966.

[24] R. L. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, 17:416–429, 1969.

[25] Sungjin Im, Nathaniel Kell, Janardhan Kulkarni, and Debmalya Panigrahi. Tight bounds for online vector scheduling. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 525–544, 2015.

[26] David S. Johnson. Fast algorithms for bin packing. *J. Comput. Syst. Sci.*, 8(3):272–314, 1974.

[27] David S. Johnson, Alan J. Demers, Jeffrey D. Ullman, M. R. Garey, and Ronald L. Graham. Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM J. Comput.*, 3(4):299–325, 1974.

[28] Adam Meyerson, Alan Roytman, and Brian Tagiku. Online multidimensional load balancing. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 16th International Workshop, APPROX 2013, and 17th International Workshop, RANDOM 2013, Berkeley, CA, USA, August 21-23, 2013. Proceedings*, pages 287–302, 2013.

[29] Michael Mitzenmacher. Load balancing and density dependent jump markov processes (extended abstract). In *37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14-16 October, 1996*, pages 213–222, 1996.

[30] Michael David Mitzenmacher. *The Power of Two Choices in Randomized Load Balancing*. PhD thesis, UC Berkeley, 1996.

[31] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1997.

[32] Yuval Peres, Kunal Talwar, and Udi Wieder. The (1 + beta)-choice process and weighted balls-into-bins. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 1613–1619, 2010.

[33] Jiri Sgall. Online scheduling. In *Algorithms for Optimization with Incomplete Information, 16.-21. January 2005*, 2005.

[34] Jirí Sgall. Online bin packing: Old algorithms and new results. In *CiE*, volume 8493 of *Lecture Notes in Computer Science*, pages 362–372. Springer, 2014.

[35] Kunal Talwar and Udi Wieder. Balanced allocations: the weighted case. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 256–265, 2007.

[36] Berthold Vöcking. How asymmetry helps load balancing. *J. ACM*, 50(4):568–589, 2003.

[37] Andrew Chi-Chih Yao. Probabilistic computations: Toward a unified measure of complexity (extended abstract). In *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977*, pages 222–227, 1977.

[38] Xiaojun Zhu, Qun Li, Weizhen Mao, and Guihai Chen. Online vector scheduling and generalized load balancing. *J. Parallel Distrib. Comput.*, 74(4):2304–2309, 2014.