Tight Bounds for Online Vector Bin Packing

Yossi Azar*
Tel-Aviv University
Email: azar@tau.ac.il

Ilan Cohen†
Tel-Aviv University
Email: ilanrcohen@gmail.com

Seny Kamara
Microsoft Research, Redmond, WA
Email: senyk@microsoft.com

Bruce Shepherd‡
McGill University, Montreal, Canada
Email: bruce.shepherd@mcgill.ca

Abstract

In the d-dimensional bin packing problem (VBP), one is given vectors $x_1, x_2, \ldots, x_n \in \mathbf{R}^d$ and the goal is to find a partition into a minimum number of feasible sets: $\{1,2\ldots,n\} = \cup_i^s B_i$. A set B_i is feasible if $\sum_{j\in B_i} x_j \leq \mathbf{1}$, where $\mathbf{1}$ denotes the all 1's vector. For online VBP, it has been outstanding for almost 20 years to clarify the gap between the best lower bound $\Omega(1)$ on the competitive ratio versus the best upper bound of $O(d)$. We settle this by describing a $\Omega(d^{1-\varepsilon})$ lower bound. We also give strong lower bounds (of $\Omega(d^{\frac{1}{B}-\varepsilon})$) if the bin size $B \in \mathbf{Z}_+$ is allowed to grow. Finally, we discuss almost-matching upper bound results for general values of B; we show an upper bound whose exponent is additively "shifted by 1" from the lower bound exponent.

1 Introduction

We study the *vector bin packing problem* (VBP) which has received renewed attention in connection with research on virtual machine placement in cloud computing, e.g., [23, 18, 22]. In the *offline* version, one is given a collection $\mathscr{V} = \{x_1, x_2, \ldots, x_n\}$ of vectors $x_i \in [0,1]^d$, and a subset $X \subseteq \mathscr{V}$ is a *feasible bin* if $\sum_{x_i\in X} x_i \leq \mathbf{1}$. If the bins have size B, then feasibility asks for $\sum_{x_i\in X} x_i \leq \mathbf{B}$ (bold face indicates a d-dimensional vector). We denote by $OPT(\mathscr{V})$ the minimum number of bins needed to partition the vectors into feasible bins. If all vectors are binary, we refer to this as $\{0,1\}$ *VBP*. We also sometimes refer to the *packing integer program* (PIP) version of the problem, where we seek the largest set of vectors which fits into a single bin.

In the *online* version, the vectors arrive one by one, and at the time of arrival, the algorithm A must either assign a vector to an open bin, or open a "new" bin to which it is assigned (maintaining bin feasibility of course). The algorithm's *competitive ratio on an instance* $\mathscr{V}$ is $\frac{A(\mathscr{V})}{OPT(\mathscr{V})}$, where $A(\mathscr{V})$ denotes the number of bins opened by A. Similarly, its competitive ratio *on a class of instances* is the worst ratio on an instance in the class; its overall competitive ratio is this measure taken on the class of all d-dimensional instances of VBP.

*Supported in part by the Israel Science Foundation (grant No. 1404/10). Part of this work was done while the author was visiting Microsoft Research, Redmond.

†Supported in part by the Google Inter-university center.

‡Supported in part by Natural Sciences Engineering Research Council of Canada

While there is a simple first-fit $O(d)$-competitive algorithm for VBP (see [11]), the best lower bounds have been constant. Specifically, the bounds from [9] are at most 2. As pointed out in [10], this gap has persisted, and in fact in [7] it is conjectured to be super-constant, though sublinear. We show (Section 2) a lower bound of $\Omega(d^{1-\varepsilon})$ for any $\varepsilon > 0$ giving an essentially tight result. The arguments make critical use of techniques from [14] for online colouring and [5] for stochastic PIPs. We also need further structural insights into the class of graphs used by [14] - see Section 2.1. Our main lower bound results are the following.

Theorem 1.1. *For any integer $B \geq 1$, any deterministic online algorithm for VBP has a competitive ratio of $\Omega(d^{\frac{1}{B}-\varepsilon})$. For $\{0,1\}$ VBP the lower bound is $\Omega(d^{\frac{1}{B+1}-\varepsilon})$.*

These lower bounds are information-theoretic and hence apply also to exponential time algorithms. These bounds are tight for $B = 1$ since there is a $(d+.7)$-competitive algorithm for VBP [11]. In fact for $B = 1$, this settles a problem pointed out in [10]: "the main open problem in on-line d-dimensional vector packing consists in narrowing the wide gap between 2 and $d + 7/10$". However, for $B \geq 2$, there was a gap in our understanding. For instance, for $\{0,1\}$ VBP and $B \geq 2$, it is known that the greedy algorithm is $2B\sqrt{d}$-competitive. This upper bound exponent of $\frac{1}{2}$ is "off" from our lower bound exponent of $\frac{1}{B+1}$ (which seems the right answer). We partially rectify this in Section 3 via the following upper bound.

Theorem 1.2. *There are online VBP algorithms for $B \geq 2$ with competitive ratio:*

- $O(d^{1/(B-1)} \log d^{B/(B-1)})$, *for* $[0,1]^d$ *vectors.*
- $O(d^{1/B} \log d^{(B+1)/B})$, *for* $\{0,1\}^d$ *vectors.*

Note that for $B \geq \log d$ the bound becomes $O(\log d)$. It is worthwhile to mention that there is still an additive"shift by 1" gap in the exponent. Specifically, for $\{0,1\}$ vectors, the exponent of the upper bound is $1/B$ whereas the exponent of the lower bound is $1/(B+1)$. For general vectors the difference in exponent is $1/(B-1)$ versus $1/B$.

1.1 Techniques

For the lower bound, a core idea that we use is a lower bound strategy for online colouring that has been around for some time [14]. They show that there are graphs with chromatic number $O(\log n)$ for which any online algorithm may use up to $n/\log n$ colours. In the online colouring model, an adversary produces a sequence of nodes $v_1, v_2, \ldots, v_n$ such that upon arrival, v_i reveals its neighbours amongst $v_1, v_2, \ldots, v_{i-1}$. Moreover, in their model the adversary is *transparent* in that it reveals the actual colour of v_i just after the online algorithm makes its choice. Strangely, this extra feature is important for our arguments to work. Similar bounds for colouring are given in [13] for a stronger model, where the graph being coloured is revealed to the algorithm at the beginning (but the adversary selects a subgraph to be coloured). We refer the reader to the surveys of online packing (and covering) [10] and [4] for further background.

For the upper bound we combine some state of the art techniques. The algorithm we present is based on two parts. The first part solves VBP using a load balancing technique with a set of virtual enlarged bins (of size $cB\log d$). The algorithm decides on the assignments using exponential weights for each virtual bin. The second part assigns vectors of each virtual bin to real bins of size B. For the second part we present two algorithms. First, a simple randomized algorithm, and we use Chernoff bounds to compute the expected number of bins. Second, a deterministic algorithm which is a de-randomization of the randomized algorithm using techniques from [1]. Combining both parts produces a VBP algorithm with the desired performance.

1.2 Related Work

VBP has classical connections to scheduling problems where jobs need to be run on machines, each with a bounded supply of some d resources (e.g., CPU time, memory etc). Such applications have been actively studied recently due to interest in cloud computing, e.g., the problem of virtual machine placement and migration, cf. [18].

Epstein [7], largely motivated by the gap for online VBP, initiated the study of d-dimensional vector packing with *variable sized bins*. In this case, one is given a collection $\mathscr{B}$ of bin capacity profiles (each a vector in $\mathbf{R}^d$), where the collection is assumed to contain $\mathbf{1}$. In that setting, they can show that for certain choices of $\mathscr{B}$, one may obtain linear lower bounds, whereas for other choices, the competitive ratio can be at most $1+\varepsilon$ (the choice of $\mathscr{B}$ depends on ε). They conjecture that in the classical case, where $\mathscr{B} = \{\mathbf{1}\}$, the competitive ratio is super-constant, but sublinear. When $d = 1$, variable-sized bin packing was introduced by Friesen et al. [8] and subsequently studied in [19].

There is also extensive work on (polytime) approximation algorithms for offline VBP cf. [3]. We mention that until recently there was a gap between $\Omega(d^{.5-\varepsilon})$ and $O(d^{1-\varepsilon})$ for the approximation version of offline PIP. This was settled in [5] in their study of stochastic PIPs. In [2] it was shown that for any $\varepsilon > 0$, a polytime $d^{.5-\varepsilon}$-approximation for VBP would imply NP=ZPP. By using techniques of [6] it is not hard to strengthen this lower bound to $d^{1-\varepsilon}$; the arguments become apparent in Section 2.2.

Proposition 1.3. *For any $\varepsilon > 0$, a polytime $d^{1-\varepsilon}$-approximation for VBP implies that NP=ZPP.*

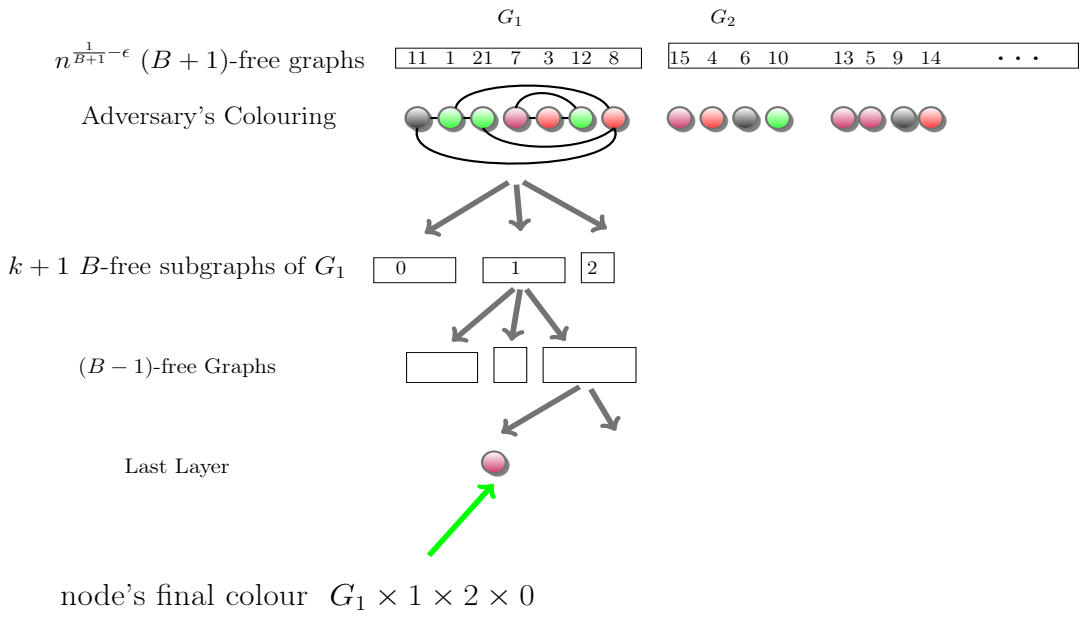
In contrast, our lower bound is information-theoretic and holds without any complexity assumptions. In particular it applies also to exponential time algorithms.

We close by remarking that while online packing (PIPs) are structurally very related to online VBP, lower bounds for the packing version are more immediate. For instance, the online maximum stable set problem (in the setting described above for online colouring) is completely trivial. This is in stark contrast to the relatively rich theory for online colouring (cf. [20, 12, 15, 17]). A trivial lower bound of n is achieved as follows. As soon as the online algorithm puts some node in its stable set, the adversary makes all future nodes adjacent to the selected node. This argument is not quite correct for online PIPs, since the adversary must reveal the whole column being packed (i.e., adjacencies to future nodes as well). See Appendix A for a slightly more elaborate and valid argument.

2 Lower Bounds

A core idea in the first step of our lower bound proof employs arguments from [14] which shows a lower bound for online colouring. Importantly for us, their results hold for the following class of graphs, which we call the *HS graphs*. An HS graph G of *order* k (assume k even) is a graph whose nodes are a sequence $(v_1, v_2, \ldots, v_n)$ with the following additional properties. G has a k-colouring $c : V \to [k]$ which is *compatible* with some *admissible colour assignment*. Each node v has been assigned a $k/2$ subset $F(v)$ of $[k]$ (its inadmissible set) and for each $u \prec v$ (in the sequence), $uv \in E(G)$ if and only if $c(u) \in F(v)$. We call this extra property *consistency*. The number of nodes in such an HS graph is assumed to be $n = \frac{k}{2} \cdot \binom{k}{\frac{k}{2}}$, and so $k = O(\log n)$. In particular, it is important to note that consistency implies: *if v is adjacent to some node of colour c before it in the sequence, then it is adjacent to all nodes of colour c before it.*

In their work, the adversary produces a sequence of n nodes $v_1, v_2, \ldots, v_n$ of some HS graph. When it announces v_i, it must share all adjacencies to earlier nodes in the sequence. The adversary can even



announce k and n ahead of time. Moreover, it "taunts" the online algorithm in the following sense: after the online algorithm colours v_i, the adversary announces its real colour $c(v_i)$. These additional properties are referred to as a *transparent* adversary. They establish that there are such graphs with $k = O(\log n)$ for which **any** online algorithm will use $\Omega(\frac{n}{\log n})$ colours. That is, the adversary may force the algorithm to use this many colours. Their elegant result is succinctly captured in the following.

Theorem 2.1 ([14])**.** *For any online colouring algorithm, its competitive ratio on the class HS is at least $2n/log^2 n$, even against a transparent adversary. Specifically, any online colouring algorithm uses at least $2n/log n$ colours, where any graph in the HS class is $log n$ coloured.*

2.1 The {0,1} Case

We now establish the second half of Theorem 1.1. This bound is tight in the $B = 1$ case, and almost-tight for $B > 1$. (See Section 3.)

Theorem 2.2. *For any fixed $\varepsilon \in (0,1)$, any online $\{0,1\}$ VBP algorithm has competitive ratio at least $d^{\frac{1}{B+1}-\varepsilon}$.*

Proof. Theorem 2.1 strongly suggests a reduction strategy to obtain lower bounds for $\{0,1\}$ VBP. We follow this approach, and show how to convert an online stream of nodes in an HS graph, into an online stream of vectors for VBP. We do this so that if there is a VBP algorithm with competitive ratio $O(d^{\frac{1}{B+1}-\varepsilon})$, then there is an online colouring algorithm with competitive ratio $O(n^{1-\varepsilon})$, thus contradicting Theorem 2.1. There are two complications. The first is to deal with the fact that in VBP, the online algorithm has complete information about incoming vectors, whereas in online colouring, we only know about adjacencies to nodes which arrived earlier. We can deal with this by expanding the dimension of the space where packing occurs from $d = n$ to $d = n^{B+1}$. Partly as a warm-up and partly for pedagogy, we describe this for the case $B = 1$. The ideas are core to the more complicated arguments for larger B.

Consider a stream of nodes arriving from an HS graph. For each new node v_i, we supply the online VBP algorithm a $0,1$ vector x_i with n 1's, and (n^2-n) 0's. These form the columns of a matrix A that is gradually built by the algorithm. For each $j<i$, if v_iv_j is an edge of the HS graph, then find a row which has exactly one 1 in column j. The vector x_i (i.e., next column of A) will also include a 1 in this row. If v_i saw r earlier nodes, then we place r 1's in this fashion. We then place $(n-r)$ 1's in fresh rows, i.e., rows such that A does not yet contain any 1's. These will be used for later nodes which are adjacent to v_i.

Let A be the final matrix constructed, and note that if $Az \leq \mathbf{1}$ for some $z \in \{0,1\}^n$, then z must be the incidence vector of a stable set. Hence the optimal offline solution for VBP is precisely $\chi(G)$, the chromatic number of G. Since the number of rows $d=n^2$, we have the following. If there is an online algorithm for $\{0,1\}$ VBP with competitive ratio at most $d^{.5-\varepsilon}$, then it would produce a colouring of G that uses at most $n^{1-2\varepsilon}$ colours. For any fixed $\varepsilon>0$, this contradicts Theorem 2.1.

Note that this actually completes the proof in the case when $B=1$. For $B\geq 2$, there are further complications. First, we use a construction from [6]. This helps us convert a stream of nodes in an HS graph, into a stream of vectors with the following property. A valid online VBP will pack these vectors into bins such that each bin corresponds to a K_{B+1}-free graph of the HS graph. The second issue is to turn this into a "good" colouring of the original graph. To each K_{B+1}-free graph, we apply a second online *filtering* algorithm to produce a good colouring. Interestingly, we make critical use of both the structure of HS graphs, as well as the fact that the bounds of [14] are with respect to a transparent adversary. We need to know the earlier colours!

Again, for simplicity we start by explaining the argument for the case $B=2$. Suppose that we have an online algorithm for $\{0,1\}$ VBP with $B=2$ which is $d^{\frac{1}{B+1}-\varepsilon}$-competitive for $\varepsilon \in (0,1)$. Consider the nodes of an HS graph G arriving $v_1,v_2,\ldots,v_n$. To each of these we associate a $0,1$ vector x_i in $\mathbf{R}^{n^3}$ with n^2 1's and (n^3-n^2) 0's. As before the vector x_i must prepare for future adjacencies. Now, it actually prepares for future triangles that contain it. For each triangle v_i,v_j,v_p involving node i, the algorithm does the following. If $i<j,p$ (it is the first node of the triangle to arrive) then it places a 1 in some row of A which does not yet contain any 1's; this is possible since there are $d=n^3$ rows. Otherwise, if say $j<i$, then there is some row which is already dedicated to this triangle and has a 1 in column j (and column p if $p<i$). We also place a 1 in this row of vector x_i. Consider the final matrix A constructed, and a $0,1$ vector z (i.e., selection of columns) such that $Az \leq \mathbf{B}$. Then, following [6], these columns identify a triangle-free subgraph of G when $B=2$ (and a K_{B+1}-free subgraph if we were preparing vectors for future $(B+1)$-cliques more generally).

Suppose that there is a $d^{\frac{1}{B+1}-\varepsilon}$-competitive algorithm for $\{0,1\}$ VBP when $B=2$. We claim that we can convert this into an $n^{1-3\varepsilon}$-competitive algorithm for colouring HS graphs, contradicting Theorem 2.1. To see this, we feed the vectors (defined above) to our online VBP algorithm that partitions them online into colour classes $G_i : i=1,2\ldots,s$ which are necessarily triangle-free. Note first that for an HS graph of order k, the minimum partition into triangle-free graphs is clearly at most k (since it is actually k-colourable). Hence, by the competitive ratio, we have $s \leq d^{\frac{1}{3}-\varepsilon}k = n^{1-3\varepsilon}k$. We now show an online algorithm which converts these triangle-free graphs into "real" colourings.

One easily proves that any triangle-free graph on N nodes, has chromatic number at most $\sqrt{N}$. However, we do not know of an online algorithm which guarantees to find a colouring of size at most $N^{1-\beta}$ for any $\beta>0$. Instead, we appeal to the special structure of HS graphs. For each i, we feed the nodes of G_i to a second online colouring algorithm.

The second online algorithm sees the nodes of G_i arrive as $w_1,w_2,\ldots,w_N$, i.e., these are the nodes corresponding to the vectors packed into triangle-free class i. The algorithm crucially depends on the fact that when it processes some w_j it actually knows the adversary's colouring on nodes before it (but not w_j itself). If some new node w_j is not adjacent to any earlier node, we assign it colour 0. Otherwise, let w_{j^*}

be the first node in G_i's sequence to which it is adjacent. Assuming a transparent adversary, we also know a colour $c^* = c(w_{j^*}) \in \{1,2,\ldots,k\}$ for this node in G. We then assign the colour c^* to w_j for our own colouring (we will actually assign w_{j^*} a different colour, possibly 0 for instance).

We now argue that this produces a valid colouring of G_i. Note that each node is either given colour 0 (and these nodes form a stable set), or it is "sinked" into the first node in the sequence $(w_1,\ldots,w_N)$ to which it is adjacent. Suppose some w_i is sinked into the node w_{j^*} which has colour c^*. In fact w_{j^*} was the first node in this sequence to get colour c^*. That is because in HS graphs, a node is adjacent to some node of colour c^* before it in the sequence, if and only if it is adjacent to every node of this colour before it in the sequence. Hence the nodes we coloured c^* form a star with center w_{j^*}. Since G_i is triangle-free, these nodes form a stable set and so we have a valid colouring. Note that we used at most $k+1$ colours to online colour G_i. In total, we have thus used $(k+1)s \leq (k+1)n^{1-3\varepsilon}k$. We may choose the order k of our HS graph large enough so that $(k+1)n^{1-3\varepsilon}k < 2n/\log n$ contradicting Theorem 2.1. (Recall $k = O(\log n)$ in HS graphs.)

We now consider the lower bound for $B \geq 3$. Here we apply a recursive argument. Consider $\{0,1\}$ VBP where we now produce a matrix with $d = n^{B+1}$ rows, one for each $(B+1)$-clique. Then a $d^{\frac{1}{B+1}-\varepsilon}$-competitive algorithm will partition G into at most $n^{1-(B+1)\varepsilon}k$ graphs, each of which is K_{B+1}-free. On each such subgraph G_i, we run the same online filtering algorithm used above on K_3-free graphs. As before, each colour class consists of a "star", i.e., a set of nodes with a common neighbhour in the subgraph G_i. This may no longer form a stable set, but it is guaranteed to be K_B-free since G_i is K_{B+1}-free. We then apply this online algorithm again, on each of the resulting K_B-free subgraphs. We call this *filtering*. Note that a node w_i again only looks at nodes w_j which arrived before it, but in addition it only considers nodes that had the same colour in all of the preceding filtering phases. Figure 2.1 gives an example of the filtering process on the first K_{B+1}-free graph G_1. After filtering $B-2$ times, we are guaranteed to end up with K_3-free classes. One further round of filtering then guarantees that the final colour classes are indeed stable sets, just as in the $B = 2$ case. This produces a colouring of G as follows. If a node was in G_i, and the colours it received in the $B-1$ filters (one for each K_t-free subgraphs, $t = B+1, B, B-1, \ldots, 3$) were $c_{B+1}, c_B, \ldots$, then it is assigned colour $G_1 \times c_{B+1} \times c_B \ldots$. Since at most $k+1$ colours are used at each layer, the total number of colours used is at most $(k+1)^{B-1}n^{1-(B+1)\varepsilon}$. Since B is a constant, we may choose the order k of our HS graph large enough so that $(k+1)^{B-1}n^{1-(B+1)\varepsilon} < 2n/\log n$ contradicting Theorem 2.1. (Recall $k = O(\log n)$ in HS graphs.) □

2.2 The [0,1] Case

The lower bound for the general case needs some modification. In particular, we employ an idea from [6] that was used in closing the approximation gap associated to the offline version of PIP.

We show that a $d^{1-\varepsilon}$-competitive algorithm for VBP would imply an $n^{1-\varepsilon}$ competitive algorithm for colouring the family of HS graphs. We follow the proof of Theorem 2.2, feeding vectors x_i to an online VBP algorithm as we receive nodes from some graph G. The vectors now live in $[0,1]^n$. We set the i^{th} component of x_i to 1, and for each $j < i$ with $v_i v_j \in E(G)$, we place $\frac{1}{n}$ in the j^{th} component. All other entries are 0. For the final matrix A, we have that $Az \leq \mathbf{1}$ for a $0,1$ vector z, implies that z is the incidence vector of a stable set. Hence a valid (online) vector packing again identifies a valid (online) colouring of G. Since $d = n$, a $d^{1-\varepsilon}$-competitive VBP algorithm, would produce a $n^{1-\varepsilon}$-competitive algorithm for colouring HS graphs, contradicting Theorem 2.1. This establishes a complete proof of the first half of Theorem 1.1 for the case $B = 1$.

In the $B = 2$ case, we now produce vectors "prepared" for future triangles as in the proof of the $\{0,1\}$ VBP, $B = 2$ case. However, we are able to do this using only $d = n^2$ dimensions (not d^3) as follows. This combines several ideas we have seen so far. Each vector will have entries in $0, 1, \frac{1}{n}$ and each row will be

associated to (at most) one edge. Suppose that v_i, v_j, v_p forms a triangle in G where $j < p < i$. When j arrives, the vector x_j is prepared for the edge jp by placing a 1 in a fresh row. When p arrives, it places a 1 in the same row. In the future, for any node i arriving which forms a triangle with j, p the vector x_i places a $1/n$ in the row corresponding to the edge jp. One easily sees again that for the resulting matrix A, any $\{0,1\}$ solution to $Az \leq \mathbf{2}$ identifies a triangle-free subgraph. (And there is an obvious extension for general B.) The rest of the proof is similar to Theorem 2.2 and we omit the details in this submission.

2.3 Randomized Online Algorithms

It is natural to ask if randomization can improve the situation for online vector bin packing. We can, however, give similar lower bounds based on results in [14] for randomized online colouring. They use the same construction as for Theorem 2.1 but appeal to Yao's Lemma [21]. That is, it is enough to exhibit a distribution of k-colourable graphs for which the expected number of colours used by any *deterministic* algorithm is at least n/k. As before, the distribution can be chosen to have support restricted to HS graphs of order k (hence $k = O(\log n)$). The adversary constructs a graph from this distribution in advance and hence *oblivious* to the choices of the online algorithm. They prove the following for sufficiently large n: *the competitive ratio of any randomized online colouring algorithm is at least least $n/(16log^2 n)$, even if its input is restricted to HS graphs.*

We may piggyback on their analysis of expected behaviour of a deterministic algorithm as follows. An HS graph is determined by a sequence of nodes v_i together with a subset $F(v_i)$ of admissible colours, where $|F(v_i)| = k/2$, and an assignment of colours to the nodes (obeying consistency). The distribution is as follows. The graph is generated iteratively where each $F(v_i)$ is a random $k/2$ subset of $[k]$. Then v_i is made adjacent to the appropriate nodes to preserve consistency. It is then assigned a random colour from $[k] - F(v_i)$ (the admissible colours). Consider $\{0,1\}$ VBP. We may define a 1-1 map to the space of $n^2 \times n$ matrices used in the lower bound of VBP. As we have argued, a solution of size X to the VBP instance immediately yields a solution of the same size for the colouring instance. Hence the expected performance of a deterministic algorithm for VBP could be no better than that for online colouring.

3 A competitive VBP algorithm

In the case of $B = 1$ our lower bounds are tight. Indeed, for the $[0,1]^d$ case, there is a $(d + .7)$-competitive algorithm based on first fit [11]. The bounds are also tight for the $\{0,1\}$ case - see Section 5.

We now present and algorithm for $B \geq 2$. As mention in the introduction, the algorithm is based on two parts. The first part solves VBP using a load balancing technique with a set of virtual enlarged bins (of size $cB \log d$). The load balancing uses exponential weights to decide on the assignment. The second part assigns vectors of each virtual bin to real various bins of size B. For the second part we present two algorithms, a randomized algorithm and a deterministic algorithm. Combining both parts produces a VBP algorithm with the desired performance.

3.1 The virtual VBP algorithm.

In the following, let $opt = OPT(x)$ be the minimum number of bins of size B required to assign vectors sequence x in off-line fashion. We present an online VBP algorithm that uses $4opt$ virtual bins with size $cB \log d$ for some constant $c > 0$. The input is an online stream of vectors $x \in \mathbf{R}^d$. Let the i^{th} vector be $x_i = (x_{i1}, \ldots, x_{id})$ where $x_{ik} \in [0,1]$ for $1 \leq i \leq n$ and $1 \leq k \leq d$. An online load balancing VBP algorithm

assigns each vector to a bin. Let $A(i) = j$ if vector i is assigned to bin j by the algorithm. The *load* of bin j in coordinate k just before vector i arrives is

$$L^i_{jk} = \sum_{i'<i:A(i')=j} x_{i'k}.$$

In the following, when we drop the superscript, then L_{jk} denotes the loads upon termination of the algorithm. Additionally, let $\tilde{x}_{ik} = x_{ik}/B$ and $\tilde{L}^i_{jk} = L^i_{jk}/B$.

First we construct a procedure for virtual bin packing for what we call d-balanced vectors. A vector $x_i \in [0,1]^d$ is *d-balanced* if it satisfies:

$$\max_{r,k,x_{ik}>0} x_{ir}/x_{ik} \leq d.$$

Let $M_i = \max_k x_{ik}$ and $c_1 > 0, 2 > a > 1$ be constants that will be determined later. We now present Procedure 1 for virtual bin packing of d-balanced vectors and Algorithm Virtual-VBP that uses Procedure 1 for virtual bin packing of general vectors.

$m \leftarrow 1$. Open one active bin;
foreach *vector* x_i **do**
  Let j be the active bin with minimal $\sum_k a^{\tilde{L}^i_{jk}+\tilde{x}_{ik}} - a^{\tilde{L}^i_{jk}}$;
  if $\forall k : \tilde{L}^i_{jk} + \tilde{x}_{ik} \leq c_1 \log d$ **then**
    assign vector i to bin j;
  else $*failure*$
    De-activate the m active bins;
    $m \leftarrow 2m$. Open and active new m active bins;
    Assign x_i to the first (empty) active bin;
end

Procedure 1: Virtual VBP for d-balanced vectors

begin
  Init Procedure 1 for simulation;
  foreach *vector* x_i **do**
    Define x'_i as:
    $$x'_{ik} = \begin{cases} 0 & \text{if } \frac{x_{ik}}{M_i} \leq \frac{1}{d}. \\ x_{ik} & \text{otherwise.} \end{cases}$$
    Use Procedure 1 to assign vector x'_i to simulated bin j;
    Assign x_i to a virtual bin j;
  end
end

Algorithm Virtual-VBP: Virtual VBP for general vectors

Theorem 3.1. *Algorithm Virtual-VBP uses at most* $4opt$ *bins of size* $cB\log d$ *(where* $c = 2c_1$*).*

First we establish some properties of Procedure 1 when assigning a sequence x of d-balanced vectors.

Lemma 3.2. *Procedure 1 never fails on d-balanced vectors when it uses $m \geq opt$ active bins of size $c_1 B \log d$.*

We will use Lemma 3.2 to prove the constant (in term of number of bins) bound.

Corollary 3.3. *When applying Procedure 1 on d-balanced vectors, Procedure 1 opens at most* $4opt$ *bins.*

of Corollary 3.3. Using Lemma 3.2, if the number of active bins $m \geq opt$, then Procedure 1 does not open any new bins. Therefore, the number of active bins is less than $2opt$ and the total number of opened bins is less than $4opt$. □

In order to prove Lemma 3.2 we first show the following two lemmas.

Lemma 3.4. $\forall a > 1$ *and* $0 \leq x \leq 1, a^x - 1 \leq (a-1)x$

Proof. The function $f(x) = (a-1)x$ is linear in the section $[0,1]$, but $g(x) = a^x - 1$ is convex. The functions intersect at $x = 0$ and $x = 1$ and the lemma follows immediately since the functions are continuous. □

Lemma 3.5. *When applying Procedure 1 on d-balanced vectors, then for any active bins j, j' and for each coordinate k' the following holds:*

$$\sum_{k=1}^{d} a^{\tilde{L}_{jk}} \geq a^{\tilde{L}_{j'k'}} \frac{\ln a}{d(a-1)a^{1/B}}.$$

Proof. Let i' be the last vector assigned to bin j' with $x_{i'k'} > 0$. Then for any active bin j:

$$\sum_{k=1}^{d} a^{\tilde{L}^{i'}_{j'k}+\tilde{x}_{i'k}} - a^{\tilde{L}^{i'}_{j'k}} \leq \sum_{k=1}^{d} a^{\tilde{L}^{i'}_{jk}+\tilde{x}_{i'k}} - a^{\tilde{L}^{i'}_{jk}}.$$

The left side of the inequality satisfies

$$\begin{aligned}
\sum_{k=1}^{d} a^{\tilde{L}^{i'}_{j'k}+\tilde{x}_{i'k}} - a^{\tilde{L}^{i'}_{j'k}} &\geq a^{\tilde{L}^{i'}_{j'k'}}(a^{\tilde{x}_{i'k'}} - 1) \\
&\geq a^{\tilde{L}^{i'}_{j'k'}} \ln(a)\tilde{x}_{i'k'} \\
&\geq a^{(\tilde{L}_{j'k'}-1/B)} \ln(a)\tilde{x}_{i'k'}
\end{aligned}$$

where the first inequality follows since the coordinate k' is part of the sum; the second inequality follows since for $x > 0$, $e^x - 1 \geq x$ ($x = \tilde{x}_{i'k'} \ln(a) > 0$ for $a > 1$); the third inequality follows since i' is the last vector assigned to bin j' and $\tilde{x}_{i'k'} \leq 1/B$.
The right side of the inequality satisfies

$$\begin{aligned}
\sum_{k=1}^{d} a^{\tilde{L}^{i'}_{jk}+\tilde{x}_{i'k}} - a^{\tilde{L}^{i'}_{jk}} &= \sum_{k=1}^{d} a^{\tilde{L}^{i'}_{jk}}(a^{\tilde{x}_{i'k}} - 1) \\
&\leq \sum_{k=1}^{d} a^{\tilde{L}_{jk}}(a-1)\tilde{x}_{i'k}.
\end{aligned}$$

The inequality follows from Lemma 3.4 and monotonicity of $a^{\tilde{L}^{i'}_{jk}}$. From the two parts we get

$$\sum_{k=1}^{d} a^{\tilde{L}_{jk}}(a-1)\tilde{x}_{i'k} \geq a^{(\tilde{L}_{j'k'}-1/B)}\tilde{x}_{i'k'} \ln a.$$

Since $x_{i'}$ is d-balanced, for all k we have $\tilde{x}_{i'k}/\tilde{x}_{i'k'} \leq d$ and therefore:

$$\sum_{k=1}^{d} a^{\tilde{L}_{jk}} \geq a^{\tilde{L}_{j'k'}} \frac{\ln a}{d(a-1)a^{1/B}}.$$

□

of Lemma 3.2. Assume there are $m \geq opt$ active bins in Procedure 1. We prove that when assigning all the vectors to the m active bins the load never exceeds $c_1 B \log d$ and hence there is no failure.

Let $j^{opt}(i)$ be the active bin in which the optimal algorithm assigned vector i. We can assume such a bin exists since $m \geq opt$. By the definition of Procedure 1 the next chosen bin j satisfies:

$$\begin{aligned}
\sum_{k=1}^{d} a^{\tilde{L}^i_{jk}+\tilde{x}_{ik}} - a^{\tilde{L}^i_{jk}} &\leq \sum_{k=1}^{d} a^{\tilde{L}^i_{j^{opt}(i)k}+\tilde{x}_{ik}} - a^{\tilde{L}^i_{j^{opt}(i)k}} \\
&= \sum_{k=1}^{d} a^{\tilde{L}^i_{j^{opt}(i)k}} (a^{\tilde{x}_{ik}} - 1) \\
&\leq \sum_{k} a^{\tilde{L}^i_{j^{opt}(i)k}} (a-1)\tilde{x}_{ik} \\
&\leq \sum_{k=1}^{d} a^{\tilde{L}_{j^{opt}(i)k}} (a-1)\tilde{x}_{ik}
\end{aligned}$$

where the second inequality follows from Lemma 3.4 and the last inequality follows from the monotonicity of the load. Summing over all vectors:

$$\sum_{i} \sum_{k=1}^{d} a^{\tilde{L}^i_{j(i)k}+\tilde{x}_{ik}} - a^{\tilde{L}^i_{j(i)k}} \leq$$

$$(a-1) \sum_{i} \sum_{k=1}^{d} a^{\tilde{L}_{j^{opt}(i)k}} \tilde{x}_{ik}.$$

By replacing the order of summation we get

$$\sum_{j=1}^{m} \sum_{k=1}^{d} \sum_{i|A(i)=j} a^{\tilde{L}^i_{jk}+\tilde{x}_{ik}} - a^{\tilde{L}^i_{jk}} \leq$$

$$(a-1) \sum_{j=1}^{m} \sum_{k=1}^{d} a^{\tilde{L}_{jk}} \sum_{i|OPT(i)=j} \tilde{x}_{ik}.$$

The load in the optimal assignment is at most B and hence $\sum_{i|OPT(i)=j} \tilde{x}_{ik} \leq 1$. Additionally, for each j,k the left hand side is a telescoping sum and hence

$$\sum_{j=1}^{m} \sum_{k=1}^{d} (a^{\tilde{L}_{jk}} - a^0) \leq (a-1) \sum_{j=1}^{m} \sum_{k=1}^{d} a^{\tilde{L}_{jk}}.$$

Therefore,

$$\sum_{j=1}^{m} \sum_{k=1}^{d} a^{\tilde{L}_{jk}} - dm \leq (a-1) \sum_{j=1}^{m} \sum_{k=1}^{d} a^{\tilde{L}_{jk}}.$$

Hence,

$$\sum_{j=1}^{m}\sum_{k=1}^{d} a^{\tilde{L}_{jk}} \leq \frac{dm}{2-a}.$$

Using Lemma 3.5, for each bin-coordinate pair j',k':

$$m \cdot a^{\tilde{L}_{j'k'}} \frac{\ln a}{d(a-1)a^{1/B}} \leq \sum_{j=1}^{m}\sum_{k=1}^{d} a^{\tilde{L}_{jk}} \leq \frac{dm}{2-a}$$

By rearranging the terms:

$$a^{\tilde{L}_{j'k'}} \leq d^2 \frac{(a-1)a^{1/B}}{(2-a)\ln a}.$$

For fixed $a \in (1,2)$ we get $\tilde{L}_{j'k'} \leq c_1 \log d$ as claimed.

□

We are now ready to prove the feasibility and competitiveness of Algorithm Virtual-VBP for general vectors.

of Theorem 3.1. Note that the number of bins opened by Algorithm Virtual-VBP on vector sequence x is the same as the number of bins opened by Procedure 1 on vector sequence x'. Recall that $x'_{ik} = x_{ik}$ if $x_{ik} > M_i/d$ and 0 otherwise. Therefore, for all i,k we have $x_{ik} \geq x'_{ik}$ hence $OPT(x') \leq OPT(x)$. By Corollary 3.3 the number of bins opened by Procedure 1 is at most $4OPT(X') \leq 4OPT(X)$. Additionally, the load on each bin is:

$$\begin{aligned}
\sum_{i:A(i)=j} x_{ik} &= \sum_{i:A(i)=j} (x_{ik} - x'_{ik}) + \sum_{i:A(i)=j} x'_{ik} \\
&\leq \sum_{i:A(i)=j} \frac{M_i}{d} + \sum_{i:A(i)=j} x'_{ik} \\
&\leq \frac{dc_1 B \log d}{d} + c_1 B \log d \\
&= 2c_1 B \log d.
\end{aligned}$$

The second inequality follows from Lemma 3.2 and from $\sum_{i:A(i)=j} M_i \leq d \cdot c_1 B \log d$ (by volume consideration).

□

3.2 The load balancing VBP algorithm.

In the last section we showed an online algorithm that packs the vectors into virtual bins of size $cB\log d$. In order to solve the VBP we need to pack into bins with size B. For that, we distribute each virtual bin's vectors into real bins. Initially we open r real bins for each virtual bin and whenever a vector is assigned to this virtual bin, we choose uniformly at random one of the real bins associated with it and assign the vector to this real bin if it is feasible. In a case of a failure, i.e. if the load of the chosen real bin after assignment exceeds B, then we open r new real bins and direct future vectors to the new real bins. By using Chernoff bounds and choosing r appropriately, the probability of failure is small enough so that the expected total number of bins associated with each virtual bin is $O(r)$.

```
begin
    Init Algorithm Virtual-VBP for simulation;
    foreach vector x_i do
        Use Algorithm Virtual-VBP to assign x_i to virtual bin j;
        Use Procedure 2 to assign vector x_i of virtual bin j to its real bin;
    end
end
```

Algorithm LB-VBP: The VBP algorithm

```
begin
    Open r new active real bins;
    foreach vector x_i assigned to this virtual bin do
        Choose uniformly at random active bin j;
        if ∀k : L^i_jk + x_ik ≤ B then
            Assign vector i to real bin j;
        else *failure*
            De-activate the active bins ;
            Open r new active bins;
            Assign x_i to a random active (empty) bin;
    end
end
```

Procedure 2: Distributing vectors from a single virtual bin (randomized)

Theorem 3.6. *Algorithm LB-VBP, using* $r = \Theta(d^{1/(B-1)} \log d^{B/(B-1)})$ *in Procedure 2, provides a feasible assignment and achieves (expected) competitive ratio of* $O(d^{1/(B-1)} \log d^{B/(B-1)})$.

In order to prove Theorem 3.6 we prove the following lemmas.

Lemma 3.7. *Given a virtual bin with size* $cB \log d$. *Distribute all of its vectors uniformly at random into* $r = (ce \log d)^{B/(B-1)}(2d)^{1/(B-1)}$ *real bins. The probability that there exists a bin coordinate whose load exceeds B is less then* $1/2$.

Proof. Let $x_1, x_2, ...x_N$ be the vectors assigned to the virtual bin. Let $Y_1^j, ..., Y_N^j$ be random variables, where Y_i^j represents whether vector i is directed to the bin j. Let $X_1^{jk}, ..., X_N^{jk}$ representing the load accumulated in bin j at coordinate k by vector i ,i.e. $X_i^{jk} = Y_i^j x_{ik}$. The load in bin j at coordinate k is $X^{jk} = \sum_i X_i^{jk}$. Notice that, $E[X_i^{jk}] = x_{ik}/r$ and $E[X^{jk}] = E[\sum_i X_i^{jk}] = \sum_i x_{ik}/r \leq cB \log d/r \leq B$. Let $\mu_k = E[X^{jk}]$. $X_i^{jk} \in [0,1]$ and therefore by a Chernoff bound for any $\delta_k > 1$:

$$Pr[X^{jk} \geq \delta_k \mu_k] \leq \left(\frac{e}{\delta_k}\right)^{\mu_k \delta_k}.$$

Let $\delta_k = B/\mu_k \geq 1$ and $\delta = \min_k \delta_k$. Therefore, the probability that overload occurs in bin j at coordinate k is

$$Pr[X^{jk} \geq B] \leq \left(\frac{e}{\delta_k}\right)^B \leq \left(\frac{e}{\delta}\right)^B.$$

Using a union-bound, the probability of an overload in some bin-pair coordinate is:

$$\begin{aligned} Pr[\exists j,k | X^{jk} \geq B] &\leq drPr[X^{jk} \geq B] \\ &\leq dr\left(\frac{e}{\delta}\right)^B \\ &\leq dr\left(\frac{ec\log d}{r}\right)^B \\ &= \frac{d(ec\log d)^B}{r^{B-1}} \\ &= \frac{1}{2}. \end{aligned}$$

where the last inequality follows from $B/\delta = \max_k \mu_k \leq cB \log d/r$ and last equality follows from the definition of r. □

Lemma 3.8. *Let* $r = (ce \log d)^{B/(B-1)}(2d)^{1/(B-1)}$. *The expected number of bins opened by the Procedure 2 is* $O(r)$.

Proof. By Lemma 3.7 using $r = (ce \log d)^{B/(B-1)}(2d)^{1/(B-1)}$ the probability of opening r additional bins is less then $1/2$. Clearly, the probability of an i^{th} additional opening of bins given $(i-1)$ additional openings is less then $1/2$. Hence, the probability for opening ir bins is less then $1/2^i$. Therefore, the expected number of open bins is at most $2r$. □

of Theorem 3.6. Since Procedure 2 checks that the load is at most B, the algorithm is feasible. From Lemma 3.1 the number of virtual bins opened is at most $4opt$, each with size $cB\log d$. From Lemma 3.8 the expected number of bins opened by each instance for these virtual bin vectors is $O(d^{1/(B-1)}\log d^{B/(B-1)})$. Therefore, the total expected number of bins is $O(opt \cdot d^{1/(B-1)}\log d^{B/(B-1)})$. □

From Theorem 3.6 we prove the existence of a randomized algorithm satisfying the first part of Theorem 1.2.

4 De-randomizing the load balancing VBP algorithm.

We now present a procedure that distributes the vectors of a single virtual bin deterministically, with up to a constant same number of bins as the randomized Procedure 2. This yields a deterministic VBP algorithm with the same competitive ratio as Algorithm LB-VBP. We use de-randomizing techniques from [1]. Our algorithm would use $Tc\log d$ bins (where T will be chosen later). We will use potential function, when a vector arrives, the algorithm assigns the vector to one of the bins, if by doing so the following potential function does not increase. We will show that in each step such a bin exists. Let $T' = \ln(T+1)$. Define the potential of coordinate k in bin j after the arrival of vector i as

$$\Phi^i_{jk} = \exp\left\{T'L^i_{jk} - \sum_{i'=1}^{i}\frac{x_{i'k}}{c\log d}\right\}$$

The potential function after the arrival of vector i is:

$$\Phi^i = \frac{1}{dTc\log d}\sum_{j=1}^{Tc\log d}\sum_{k=1}^{d}\Phi^i_{jk}$$

begin
  Open new $Tc\log d$ active bins;
  foreach *vector x_i assigned to this virtual bin* **do**
    Assign vector i to bin j s.t. $\Phi^i \leq \Phi^{i-1}$.
  end
end

Procedure 3: Distributing single virtual bin (de-randomized)

First we prove the correctness of the Procedure 3, we prove that in each step such bin always exists.

Lemma 4.1. *Initially, $\Phi^0 \leq 1$ and throughout the algorithm, $\Phi^i > 0$. In addition, when vector i arrives, there exists a bin j such that assigning vector i to bin j does not increase the potential function.*

Proof. It is easy to verify that initially the value of the potential function is exactly 1. Also, since the potential function is a sum of exponential functions it is always positive. To prove the second part of the claim we use a probabilistic argument. When vector i arrives we choose uniformly at random one of the bins and assign the vector to it, i.e. each bin is selected with probability $1/(Tc\log d)$. We prove that with this random trial the expected value of the potential function does not increase. This means that there exists a bin j such that assigning the vector i to j does not increase the potential function. The expected value of Φ^i_{jk} is:

$$
\begin{aligned}
E[\Phi^i_{jk}] &= \frac{1}{Tc\log d}\left(\Phi^{i-1}_{jk}e^{T'x_{ik}-\frac{x_{ik}}{c\log d}}\right) \\
&+ \left(1-\frac{1}{Tc\log d}\right)\left(\Phi^{i-1}_{jk}e^{-\frac{x_{ik}}{c\log d}}\right) \\
&= \Phi^{i-1}_{jk}e^{-\frac{x_{ik}}{c\log d}}\left(\frac{1}{Tc\log d}e^{T'x_{ik}}+1-\frac{1}{Tc\log d}\right) \\
&= \Phi^{i-1}_{jk}e^{-\frac{x_{ik}}{c\log d}}\left(\frac{1}{Tc\log d}(e^{T'x_{ik}}-1)+1\right) \\
&\leq \Phi^{i-1}_{jk}e^{-\frac{x_{ik}}{c\log d}}\left(\frac{1}{Tc\log d}(e^{T'}-1)x_{ik}+1\right) \\
&= \Phi^{i-1}_{jk}e^{-\frac{x_{ik}}{c\log d}}\left(\frac{x_{ik}}{c\log d}+1\right) \\
&\leq \Phi^{i-1}_{jk}
\end{aligned}
$$

where the first inequality from Lemma 3.4 for $a = e^{T'}$ and $x = x_{ik}$, the last equality follows from the definition of T', the last inequality follows from for any $x > 0$ we have $1 + x \leq e^x$. By linearity of expectation we get $E[\Phi^i] \leq \Phi^{i-1}$. □

Lemma 4.2. *Given a virtual bin with size $cB\log d$. Assigning its vectors using Procedure 3 to $Tc\log d$ bins for $T' = (\ln(dc\log d)+B)/(B-1)$ (recall that $T = e^{T'}-1$) then the load on each bin coordinate is at most B.*

Proof. From Lemma 4.1 the potential $\Phi \leq 1$, since the potential is the summation of positive terms then for all j,k,i we have $\Phi^i_{jk}/(Tc\log d) < 1$. Hence

$$exp\{T'L^i_{jk} - \sum_{i'=1}^{i}\frac{x_{i'k}}{c\log d}\} < dTc\log d$$

By rearranging the terms

$$
\begin{aligned}
L^i_{jk} &< \frac{\ln(dTc\log d)+\sum_{i'=1}^{i}\frac{x_{i'k}}{c\log d}}{T'} \\
&\leq \frac{\ln(dTc\log d)+B}{T'} \\
&\leq \frac{T'+\ln(dc\log d)+B}{T'} \\
&= B
\end{aligned}
$$

where the second inequality follows from
$\sum_i x_{ik} \leq cB\log d$ as these are vectors that assigned to a virtual bin, the last inequality implied by the definition of T and the last equality implied by the definition of T'. □

To get a deterministic *VBP* algorithm, we modify Algorithm LB-VBP, by replacing Procedure 2 with Procedure 3. From these two lemmas above we get that this algorithm is feasible and has competitive ratio of
$O(d^{1/(B-1)} \log d^{B/(B-1)})$.

5 The VBP algorithm for the {0,1} case.

The greedy algorithm. For the $\{0,1\}$ case with $B = 1$, the greedy algorithm is known to give a $O(\sqrt{d})$-competitive algorithm. For completeness we give the analysis and show that it extends also to the $B \geq 2$ case. The algorithm greedily assigns an arriving vector to a bin with available space. If there is none, we open a new bin. We classify vectors as *large* if $\mathbf{1}^T x_i \geq \sqrt{d}$, otherwise *small*. Let the number of large vectors be k and the number small is $n-k$. Clearly $opt \geq \frac{k}{\sqrt{d}}$. Now consider a small vector x_i that is blocked. Let $B_1, B_2, \ldots, B_s$ be the bins opened when it arrived. For each $j = 1, 2, \ldots, d$ we say x_i is *j-blocked* for bin B_p, if $x_{ij} = 1$ and there is some vector $v \in B_p$ with $v_j = 1$. Let N_j be the number of j-blocked bins, then clearly $opt \geq N_j$. Moreover, since x_i was small, we have $N_j > 0$ for at most $\sqrt{d}$ values of j. Since every B_p is j-blocked for some j, we have that $s \leq \sqrt{d} N_{max} \leq \sqrt{d} opt$. It follows that the number of bins open at any time is at most $2\sqrt{d} opt$. Note that this also works for general B. In that case, if there are k large vectors, then $opt \geq \frac{k}{B\sqrt{d}}$. Now, we say we are j-blocked in a bin if $x_{ij} = 1$ and that bin already contains B vectors with a 1 in component j. If we are j-blocked by N_j bins, then again $opt \geq N_j$. Hence $s \leq \sqrt{d} N_{max} \leq \sqrt{d} opt$ for an overall bound of $2B\sqrt{d} opt$.

Improving the exponent for $B \geq 2$. For the $\{0,1\}$ case with $B \geq 2$, we modify Algorithm LB-VBP by using a different number of real bins in its procedure. Specifically, we use $O(d^{1/B} \log d^{(B+1)/B})$ real bins for each single virtual bin. This results in *VBP* algorithm for the $\{0,1\}$ case that has a competitive ratio of $O(d^{1/B} \log d^{(B+1)/B})$.

We observe that in the $\{0,1\}$ case overload does not occur if $L_{jk} < B+1$. By modifying Lemma 3.7, we prove that distribute a virtual bin vectors uniformly at random into $r = (ce \log d)^{(B+1)/B} (2d)^{1/B}$ real bins then the probability that there exists a bin coordinate whose load exceeds $B+1$ is less then $1/2$. For that, we choose $\delta_k = (B+1)/\mu_k$ and we get:

$$\begin{aligned} Pr[\exists j,k | X^{jk} \geq B+1] &\leq dr Pr[X^{jk} \geq B+1] \\ &\leq dr \left(\frac{e}{\delta}\right)^{B+1} \\ &\leq dr \left(\frac{ec \log d}{r}\right)^{B+1} \\ &= \frac{d(ec \log d)^{B+1}}{r^B} \\ &= \frac{1}{2}. \end{aligned}$$

Now, we can use Procedure 2 with $r = O(d^{1/B} \log d^{(B+1)/B})$. Therefore, the competitive ratio of the modified Algorithm LB-VBP for the $\{0,1\}$ case is $O(d^{1/B} \log d^{(B+1)/B})$. Accordingly, we can get a deterministic algorithm with the same competitive ratio by using $T' = (\ln(dc \log d) + B)/B$ in Procedure 3. The choice of this T' in Lemma 4.2 implies that $L^i_{jk} < B+1$, which prove feasibility for $\{0,1\}$ vectors and yields the same competitive ratio.

6 Open Questions

The most interesting open problem is to close the small but intriguing gap between the upper and lower bounds for general bins $B \geq 2$. In particular for the case $B = \Theta(\log d)$, there is still a gap between a logarithmic upper bound to a constant lower bound. Another interesting question arises from the filtering algorithm employed in the proof of Theorem 2.2 for online colouring K_3-free HS graphs. Namely, that algorithm needed the structure of HS graphs. Can this be eliminated? Concretely, is there an online colouring algorithm for the class of K_3-free graphs whose competitive ratio is $O(n^{1-\beta})$ for some $\beta > 0$? While such graphs are $\sqrt{n}$-colourable, it appears prudent (necessary?) to use K_3-freeness more directly. This is suggested by the fact that there is no $n^{1-\beta}$-competitive algorithm for colouring k-colourable graphs [16, 17] (indeed there is barely a sublinear bound). One approach might be to leverage the polylog competitive algorithm of [17] for colouring bipartite graphs. There are natural extensions of the question to the class of K_B-free graphs for larger B. Finally, the upper bounds are not quite tight in their exponents; this remains a challenge to explore.

7 Acknowledgements

This work was partly performed while the first and third authors were visiting Microsoft Research's Theory Group during the summer of 2011. The third author also acknowledges the kind hospitality of his host James R. Lee while visiting the CSE Department at the University of Washington during 2012, and acknowledges support from the NSERC Discovery and Accelerator Programmes.

A Easy lower bound argument for Online Packing Version

The lower bound for online stable set mentioned in Section 1.2 can be adapted easily to online PIP as follows. The adversary feeds a sequence of nodes (vectors including all past and future adjacencies) $v_1, v_2, \ldots v_i$ each of which is adjacent to nodes $v_{n/2+1}, v_{n/2+2}, \ldots, v_n$ until either (a) we reach $i = n/2$ or (b) the algorithm selects some v_i. In Case (a), the adversary then feeds the remaining nodes $v_j : j > n/2$ to form a clique. Hence the algorithm missed the $n/2$ stable set $v_j : j \leq n/2$, and can produce at best a stable set of size 1 here on in. In Case (b), the adversary will feed $v_{i+1}, \ldots, v_{n/2}$ as a clique so the algorithm may pick at most one of these. The adversary then feeds $v_j : j > n/2$ as a stable set, but then the algorithm cannot pick any of these as it committed to v_i.

References

[1] N. Buchbinder and J. Naor. Online primal-dual algorithms for covering and packing problems. *13th Annual European Symposium on Algorithms - ESA 2005*, 2005.

[2] C. Chekuri and S. Khanna. On multi-dimensional packing problems. *SIAM journal on computing*, 33(4):837–851, 2004.

[3] E.G. Coffman Jr, M.R. Garey, and D.S. Johnson. Approximation algorithms for bin packing: A survey. In *Approximation algorithms for NP-hard problems*, pages 46–93. PWS Publishing Co., 1996.

[4] J. Csirik and G. Woeginger. On-line packing and covering problems. *Online Algorithms*, pages 147–177, 1998.

[5] B.C. Dean, M.X. Goemans, and J. Vondrák. Adaptivity and approximation for stochastic packing problems. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 395–404. Society for Industrial and Applied Mathematics, 2005.

[6] B.C. Dean, M.X. Goemans, and J. Vondrák. Approximating the stochastic knapsack problem: The benefit of adaptivity. *Mathematics of Operations Research*, 33(4):945–964, 2008.

[7] L. Epstein. On variable sized vector packing. *Acta Cybernetica*, 16(1):47–56, 2003.

[8] D.K. Friesen and M.A. Langston. Variable sized bin packing. *SIAM journal on computing*, 15:222, 1986.

[9] G. Galambos, H. Kellerer, and G.J. Woeginger. A lower bound for on-line vector-packing algorithms. *Acta Cybernetica*, 11(1-2):23–34, 1993.

[10] G. Galambos and G.J. Woeginger. On-line bin packing - a restricted survey. *Mathematical Methods of Operations Research*, 42(1):25–45, 1995.

[11] MR Garey, RL Graham, DS Johnson, and A.C.C. Yao. Resource constrained scheduling as generalized bin packing. *Journal of Combinatorial Theory, Series A*, 21(3):257–298, 1976.

[12] A. Gyárfás and J. Lehel. On-line and first fit colorings of graphs. *Journal of Graph Theory*, 12(2):217–227, 1988.

[13] M.M. Halldórsson. Online coloring known graphs. In *Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*, pages 917–918. Society for Industrial and Applied Mathematics, 1999.

[14] M.M. Halldórsson and M. Szegedy. Lower bounds for on-line graph coloring. In *Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms*, pages 211–216. Society for Industrial and Applied Mathematics, 1992.

[15] S. Irani. Coloring inductive graphs on-line. *Algorithmica*, 11(1):53–72, 1994.

[16] H. Kierstead. On-line coloring k-colorable graphs. *Israel Journal of Math*, 105:93–104, 1998.

[17] L. Lovasz, M. Saks, and WT Trotter. An on-line graph coloring algorithm with sublinear performance ratio. *DISCRETE MATH.*, 75(1):319–325, 1989.

[18] R. Panigrahy, K. Talwar, L. Uyeda, and U. Wieder. Heuristics for vector bin packing.

[19] S.S. Seiden, R. Van Stee, and L. Epstein. New bounds for variable-sized online bin packing. *SIAM Journal on Computing*, 32(2):455–469, 2003.

[20] S. Vishwanathan. Randomized online graph coloring. *Journal of algorithms*, 13(4):657–669, 1992.

[21] A.C.C. Yao. Probabilistic computations: Toward a unified measure of complexity. In *18th Annual Symposium on Foundations of Computer Science*, pages 222–227. IEEE, 1977.

[22] Y.O. Yazir, C. Matthews, R. Farahbod, S. Neville, A. Guitouni, S. Ganti, and Y. Coady. Dynamic resource allocation in computing clouds using distributed multiple criteria decision analysis. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 91–98. Ieee, 2010.

[23] Q. Zhang, L. Cheng, and R. Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1(1):7–18, 2010.