Tight Bounds for Clairvoyant Dynamic Bin Packing

Yossi Azar
Tel-Aviv University
Tel-Aviv, Israel
azar@tau.ac.il

Danny Vainstein
Tel-Aviv University
Tel-Aviv, Israel
danvainstein@tau.ac.il

ABSTRACT

In this paper we focus on the Clairvoyant Dynamic Bin Packing (DBP) problem, which extends the classical online bin packing problem in that items arrive and depart over time and the departure time of an item is known upon its arrival. The problem naturally arises when handling cloud-based networks. We focus specifically on the MinUsageTime cost function which aims to minimize the overall usage time of all bins that are opened during the packing process. Earlier work has shown a $O(\frac{\log \mu}{\log \log \mu})$ upper bound where μ is defined as the ratio between the maximal and minimal durations of all items. We improve the upper bound by giving an $O(\sqrt{\log \mu})$-competitive algorithm. We then provide a matching lower bound of $\Omega(\sqrt{\log \mu})$ on the competitive ratio of any online algorithm, thus closing the gap with regards to this problem. We then focus on what we call the class of aligned inputs and give a $O(\log \log \mu)$-competitive algorithm for this case, beating the lower bound of the general case by an exponential factor. Surprisingly enough, the analysis of our algorithm that we present, is closely related to various properties of binary strings.

CCS CONCEPTS

•Theory of computation → Scheduling algorithms; Online algorithms;

KEYWORDS

Online Algorithms; Dynamic Bin Packing; Clairvoyant Setting; Competitive Ratio; Analysis of Algorithms

ACM Reference format:
Yossi Azar and Danny Vainstein. 2017. Tight Bounds for Clairvoyant Dynamic Bin Packing. In *Proceedings of SPAA'17, July 24–26, 2017, Washington, DC, USA, ,* 10 pages.
DOI: http://dx.doi.org/10.1145/3087556.3087570

1 INTRODUCTION

The classical online bin packing problem has been widely researched [11] [14]. In this problem, items arrive in an online fashion, remain permanently, and must be assigned to bins upon their arrival (without delay). Furthermore, the goal is to minimize the maximum number of bins opened during the packing process. Later, motivated by practical applications (such as cloud computing), the dynamic version of the problem has been introduced and studied [6] [2] [3], namely, the Dynamic Bin Packing (DBP) problem. In this version, items arrive in an online fashion. In addition, items not only arrive but also depart over time. The definition of this variant gives rise to two different settings, i.e., clairvoyant and non-clairvoyant. In the non-clairvoyant setting, the departure time of each item is unknown until its departure, whereas in the clairvoyant setting, the item's departure time is revealed to the online algorithm upon its arrival.

Traditionally one used the goal function of minimizing the maximum number of bins opened throughout the entire process and compare the online algorithm's performance with that of an optimal algorithm. An alternative approach is to use the momentary goal function. In this goal function the online algorithm is compared to the optimal algorithm at every moment with respect to its number of opened bins, i.e., the goal function is defined as the maximal possible (at any moment in time) ratio between the number of bins opened by the online algorithm at that time and the number of bins opened by the optimal algorithm at that same time. In the classical online bin packing problem, since once items arrive they remain permanently, these goal functions collide. Unfortunately, both goal functions fail to distinguish between the case where the online algorithm's cost function is high throughout the entire process and the case where the online algorithm's cost function is only momentarily high and low throughout the rest of the process (in both cases the optimal algorithm's cost function remains low throughout the entire process). Therefore, the MinUsageTime goal function was introduced [7] [9], which better captures the total performance of online algorithms in the dynamic case.

The newly introduced MinUsageTime goal function aims to minimize the total accumulated time of all open bins, throughout the entire packing process [7] [9] (which can also be viewed as the total energy used by the algorithm). This is a problem that typically arises in cloud-based networks. Users apply to use a server bandwidth for a certain period of time, and the aim is to assign users to servers such that the overall time the servers are functioning is minimized, while maintaining the invariant that the overall requested bandwidth applied for by the users does not exceed the server's overall bandwidth. Thus, looking at the users as items and servers as bins, a natural formalization of this problem is the MinUsageTime DBP.

In our paper we focus on the clairvoyant version of this problem. This variant also arises in real-life applications, such as cloud gaming. In such applications, the users' server-time requests can be accurately predicted upon their arrival [8]. This problem was

Supported in part by the Israel Science Foundation (grant No. 1506/16), by the I-CORE program (Center No.4/11) and by the Blavatnik Fund.

SPAA'17, July 24–26, 2017, Washington, DC, USA

978-1-4503-4593-4/17/07…$15.00
DOI: http://dx.doi.org/10.1145/3087556.3087570

defined and researched by Ren *et al*. [10]. In [10], they present a $O(\frac{\log \mu}{\log \log \mu})$-competitive algorithm where μ is defined as the ratio between the maximal and minimal lengths of all items. Furthermore, they show a constant lower bound of $\frac{1+\sqrt{5}}{2}$.

Shalom *et al*. [12] researched a similar problem, namely, the online version of interval scheduling with bounded parallelism. In this problem interval jobs arrive in an online manner and are assigned to machines such that each machine may treat a bounded number of jobs at any given moment. The above problem differs from our MinUsageTime DBP in that all jobs have the same resource demand of $1/g$ for some given g. In [12] a lower bound of g was described. Although they did not explicitly write as much, by choosing appropriate parameters in their analysis one can get a non-constant lower bound for our settings.

A natural question would be to ask whether these upper or lower bounds are tight. We show that in fact, neither is tight. Specifically, we provide an algorithm which combines the First-Fit approach with the Classify-by-Duration approach, to produce a matching upper bound of $O(\sqrt{\log \mu})$ (even when our performance is compared with an optimal algorithm that may repack items at any moment), improving quadratically upon the result given in [10]. As stated above, we also provide a new approach towards establishing a matching lower bound of $\Omega(\sqrt{\log \mu})$ (where the optimal algorithm, to which we compare our performance, may not repack items) which is strictly greater than the bound that can be deduced from [12]. Thus, effectively closing the gap on the competitive ratio of this problem.

To beat the given lower bound, we turn our focus to an interesting class of input, namely, aligned inputs. In this class of input, items of length $\in (2^{i-1}, 2^i]$ may only arrive at multiples of 2^i (i.e. $c \cdot 2^i$ for some $c \in \mathbb{N}$). Note that this may also be viewed as the following: items need to be assigned to a specific bin by the online algorithm at their arrival. However they only start taking up bin space at their appropriate binary times ($c \cdot 2^i$ for some $c \in \mathbb{N}$). We provide an $O(\log \log \mu)$-competitive algorithm, which beats the former lower bound by an exponential factor.

We also note the difference between the Non-Clairvoyant setting and the Clairvoyant one. As opposed to the Clairvoyant case, in the Non-Clairvoyant case, the best competitive ratio for a deterministic algorithm one could hope to achieve is μ which was shown via a lower bound by Li *et al*. [7]. On the other hand, the First-Fit algorithm was shown to perform nearly optimally, achieving a competitive ratio of $\mu + 4$ [13].

Contributions of this paper In this paper, we provide the following results.

- In the Clairvoyant setting, we present a hybrid algorithm that combines the first-fit and classify-by-duration strategies. By reducing the problem to inputs that end at binary times (2^i for some i, depending on the item's duration), we show that the above algorithm achieves a competitive ratio of $O(\sqrt{\log \mu})$. We note that this algorithm does not assume early knowledge of μ, but rather adapts as μ increases.
- We provide a matching lower bound of $\Omega(\sqrt{\log \mu})$ on the competitive ratio of any online algorithm (in the Clairvoyant setting), thus closing the gap for this problem.
- We beat the lower bound by focusing on an interesting family of inputs, namely, aligned input (once again in the Clairvoyant setting). In this type of input, if the item is of duration $\in (2^{i-1}, 2^i]$, it may only arrive at multiples of 2^i (i.e. $c \cdot 2^i$ for some $c \in \mathbb{N}$).
  We provide an $O(\log \log \mu)$-competitive algorithm, which indeed beats the former lower bound by an exponential factor.

Table 1 summarizes the results known in the various settings regarding MinUsageTime DBP.

		Upper Bound	Lower Bound
Clairvoyant	**General Inputs**	$O(\sqrt{\log \mu})$ [*]	$\Omega(\sqrt{\log \mu})$ [*]
	Aligned Inputs	$O(\log \log \mu)$ [*]	$\Omega(1)$
Non-Clairvoyant	**General Inputs**	$\mu + 4$ [13]	μ [7]

Table 1: A summary of the asymptotic upper and lower bounds on the competitive ratios of deterministic algorithms in the various settings regarding MinUsageTime DBP. [*] These results are shown in the following chapters.

Related Work The classical online bin packing problem was introduced by Johnson *et al*. [5] and has since been extensively researched. The problem is defined such that items of sizes in $[0, 1]$ are released in an online manner, and these items must be packed into bins while maintaining the invariant that each bin packs at most a load of 1. Our goal under this setting is to minimize the number of bins used. For this problem, the currently best known algorithm is presented by Heydrich *et al*. [4] and has a competitive ratio of roughly 1.58. On the other hand, the current best known lower bound for any deterministic algorithm has been shown to be about 1.54 [1].

Later, the DBP problem was introduced by Coffman *et al*. [6] (which will be later termed as the Non-Clairvoyant DBP problem). This problem generalizes the online bin packing problem such that each item has a departure time as well, which is revealed to the online algorithm only at its departure. Under this setting the goal function remains the same as in the classical online bin packing problem. The current best known algorithm [6] achieves a competitive ratio of 2.788. On the other hand, the current best known lower bound for any deterministic algorithm is 2.666 [15].

Both of the above problems aim to minimize the overall number of bins used. A natural extension would be to look at a different type of goal function, namely, the momentary goal function. In this goal function, the competitive ratio of the online algorithm is defined as the maximum ratio between the number of bins opened by the online algorithm, compared to the number of bins opened by the optimal algorithm, where the maximum is taken over any moment.

Later, a new type of problem was introduced by Li *et al*. [7] [9], namely, the MinUsageTime DBP problem. The goal in this problem is to minimize the overall usage time of all bins being used in the packing process. For this problem, the current best known upper bound is achieved by the First-Fit algorithm and is shown to have a competitive ratio of $\mu + 4$ [13], μ being the max/min item interval length ratio. Furthermore, a lower bound of μ with respect to the competitive ratio of any online algorithm, has been shown in [7].

More recently, Ren *et al.* [10] introduced a variant of this problem where the departure time of an item is revealed to the online algorithm at the time of its arrival. This was termed the Clairvoyant DBP problem, whereas the former problem may sometimes be referred to as the Non-Clairvoyant DBP problem.

For the Clairvoyant DBP problem, the best known upper bound was $O(\log \mu)$ when μ is not known to the online algorithm, and $\min_{n \geq 1} \mu^{\frac{1}{n}} + n + 3 = O(\frac{\log \mu}{\log \log \mu})$, otherwise [10]. On the other hand, as stated earlier, a non constant lower bound exists which appears implicitly in [12]. In this paper, we improve the upper bound to $O(\sqrt{\log \mu})$ even if μ is not known to the online algorithm, and give a matching lower bound of $\Omega(\sqrt{\log \mu})$. Furthermore, we focus on a specific set of aligned inputs (to be formally defined later) which we feel capture the essence of the problem and show a $O(\log \log \mu)$-competitive algorithm for this case.

Techniques We give an overview of the two algorithms presented in our paper, the Hybrid Algorithm (HA) and the Classify-by-Duration-First-Fit (CDFF) algorithm.

- There are two natural strategies one would try using under this setting. The first is to apply the First-Fit strategy, but this is known to be at least $\Omega(\mu)$-competitive. The other is to classify items by their duration, but this is typically as worse as $\Omega(\log \mu)$-competitive (we note that this approach may be tweaked in order to give a $\theta(\frac{\log \mu}{\log \log \mu})$-competitive algorithm). Surprisingly, HA combines the two strategies starting with First-Fit and temporarily switching when needed to the classify-by-duration strategy, to achieve a tight upper bound of $O(\sqrt{\log \mu})$. Specifically, at any moment the algorithm holds two types of bins - general (GN) type bins and classify-by-duration (CD) type bins. Upon arrival of an item, HA looks at the overall load this type of items currently contribute. If the overall load is under a defined threshold, pack this item into any GN-type bin in a First-Fit manner. Otherwise, pack this item into a CD-type bin that holds this specific type. Packing the item into a CD-type bin is also done in a First-Fit manner, however in this case, only CD-type bins that hold this specific type, are considered [1].

  Upon analysing the competitive ratio of our algorithm we defined a reduction on inputs that delays each items' departure time resulting in a moderately small number of different item types such that items of the same type either depart together or do not intersect. We then apply the reduction only to the optimal algorithm (the online algorithm remains oblivious to the reduction) and show the desired competitive ratio.
- CDFF gives a $O(\log \log \mu)$ upper bound for aligned inputs. Given a collection of items, σ, at any moment, t, CDFF classifies all items based on their lengths. Furthermore, CDFF maintains rows of bins for each type of items. However, instead of statically packing types into rows, at any given moment different rows are considered for different types, i.e., items of a specific type are packed into different rows according to their arrival time. This adaptation is what ultimately improves the competitive ratio by an exponential factor. Surprisingly enough, the analysis of CDFF's performance is done by observing various properties of random binary strings which ultimately results in an upper bound of $O(\log \log \mu)$.

  Upon analysing the competitive ratio of CDFF, we first evaluate the algorithm's performance on a very structured input and then relate any other aligned input to it. We then apply the reduction (which was defined earlier) only to the optimal algorithm and show the desired competitive ratio.

[1]We note that using any Any-Fit approach towards packing items into the GN-type bins or the CD-type bins will work just as well.

2 NOTATIONS AND PRELIMINARIES

We first formally define the Clairvoyant MinUsageTime DBP problem.

The input consists of an infinite set of bins, $\{b_1, b_2, \ldots\}$, and a set of items $\sigma = \{r_1, \ldots, r_{|\sigma|}\}$. Each item is associated with an arrival time, t_r, and departure time, f_r. Therefore, each item is associated with a time interval for which it is active, denoted by $I(r) = [t_r, f_r]$. We sometimes denote t_r by $I(r)^-$ and f_r by $I(r)^+$. We also denote the interval length by $l(I(r))$. We further associate each item with a size, denoted by $s(r) \in [0, 1]$. For a given σ, let μ denote the max/min item interval length ratio, meaning, $\mu = \max_{r \in \sigma} l(I(r)) / \min_{r \in \sigma} l(I(r))$. Furthermore, let $d(\sigma)$ denote the total space-time demand of all items in σ, namely, $d(\sigma) = \sum_{r \in \sigma} s(r) \cdot l(I(r))$. Finally, let $span(\sigma) = l(\cup_{r \in \sigma} I(r))$, denote the time during which at least one item in σ is active.

The online algorithm has to pack each item into a single bin at the time of its arrival. Furthermore, the online algorithm has no knowledge of items which have not yet arrived. Nevertheless, at the time of an item's arrival, the online algorithm will also know its departure time (since we are studying the clairvoyant setting). We further assume that the online algorithm cannot repack items, i.e., once it has assigned an item to a bin, it cannot be moved to a different bin. We define the online algorithm's goal function as its MinUsageTime and denote it as $\text{ON}(\sigma)$, i.e., $\text{ON}(\sigma) = \sum_{i=1}^{\infty} span(\{r : r \text{ was assigned to } b_i\})$. We note that we may assume w.l.o.g. that once all items from a given bin depart, that bin is considered closed and never used again.

We define two types of optimal algorithms, repacking and non-repacking, to which we compare our online algorithm's performance. An optimal repacking algorithm is defined as the optimal offline algorithm (in the sense that it sees the entire input together) that may repack items at any moment. An optimal non-repacking algorithm is defined as the optimal offline algorithm that may never repack items. Denote the former as OPT_R and the latter as OPT_{NR}. Given an input, σ, we denote OPT's number of open bins at moment t as $\text{OPT}^t(\sigma)$ and the total load of all active items at moment t as $S_t(\sigma)$.

We characterize an online algorithm's performance by its *competitive ratio*. We say that an online algorithm, ON, is $c - competitive$ for $c \geq 1$, if there exists a constant, $b \geq 0$, such that for any input, σ, $\text{ON}(\sigma) \leq c \cdot \text{OPT}(\sigma) + b$ for a given optimal algorithm.

We define a First-Fit algorithm in the usual way. Given a set of open bins and an incoming item to be packed, we pack the item into the earliest opened bin. If no bins are open, we open a new bin. Furthermore, once all items depart from a given a bin, that bin is closed and never used again.

Lastly, we give the definition on an aligned input.

Definition 2.1. We define an aligned input, σ, as follows: Items of length $\in (2^{i-1}, 2^i]$ may only arrive at multiples of 2^i, i.e., $c \cdot 2^i$ for $c \in \mathbb{N}$. We note that all items that arrive at time t, arrive with some arbitrary order, meaning that our on-line algorithm must handle each item before the next arrives.

3 UPPER BOUND OF $O(\sqrt{\text{LOG}\,\mu})$

Throughout this section we assume that the shortest item's interval is at least 1. Furthermore we assume that the input items form a continuous interval of active items (otherwise we apply our algorithm to each such interval individually).

We first give a few basic observations. Recall that given an input, σ, $\text{OPT}_R^t(\sigma)$ was defined as the number of bins OPT has open at moment t and $S_t(\sigma)$ as the overall load of all active items at moment t. Thus we get the following bound,

- $\text{OPT}_R(\sigma) = \int_{\cup_{r\in\sigma} I(r)} \text{OPT}_R^t(\sigma)\,\text{dt} \geq \int_{\cup_{r\in\sigma} I(r)} \lceil S_t(\sigma) \rceil\,\text{dt}$.

Furthermore, the following two bounds hold,

- the $time-space$ bound: $\text{OPT}_R(\sigma) \geq d(\sigma)$.
- the $span$ bound: $\text{OPT}_R(\sigma) \geq span(\sigma)$.

The $time-space$ bound holds since $d(\sigma) = \int_{\cup_{r\in\sigma} I(r)} S_t(\sigma)\,\text{dt}$. The *span* bound holds since at least one bin must be opened whenever an item is active. Both bounds have been introduced in earlier work [10] [7] [9].

Given the above two bounds a natural question would be to ask whether these bounds are tight. The following lemma shows that this is indeed the case.

Lemma 3.1. *For any sequence of items, σ,*

(1) $OPT_R(\sigma) \leq \int_{t\in span(\sigma)} 2\lceil S_t(\sigma)\rceil dt$.
(2) $OPT_R(\sigma) \leq 2 \cdot d(\sigma) + 2 \cdot span(\sigma)$.

Proof. Let $\text{OPT}_R^t(\sigma)$ denote the number of bins OPT has open at time t. In particular,

$$\int_t \text{OPT}_R^t(\sigma)\text{dt} = \text{OPT}_R(\sigma). \tag{1}$$

Since OPT may repack at any moment we can assume that the overall load of any two of OPT's bins is strictly greater than 1 (otherwise OPT may pack the items of both bins into 1 at time t and then repack them as before for time $t' > t$, only improving upon its cost function).

Let $\text{OPT}_R^t(\sigma) = n$ and let $d_1, \ldots, d_n$ denote the loads in OPT's bins at time t and $S_t(\sigma)$ denote the overall load of all active items at time t. Therefore,

$$\text{OPT}_R^t(\sigma) = n < 2\sum_{i=1}^{n} d_i \leq 2\lceil S_t(\sigma)\rceil, \tag{2}$$

where the first inequality follows from the fact that the load of any two bins is strictly greater than 1. By (1) and (2), we get,

$$\text{OPT}_R(\sigma) = \int_t \text{OPT}_R^t(\sigma)\text{dt} \leq \int_t 2\lceil S_t(\sigma)\rceil\text{dt}.$$

Therefore,

$$\begin{aligned}\text{OPT}_R(\sigma) &\leq \int_t 2\lceil S_t(\sigma)\rceil\text{dt} \leq \int_t 2 + S_t(\sigma)\text{dt} \\ &= 2 \cdot span(\sigma) + 2 \cdot d(\sigma),\end{aligned}$$

which gives us the desired results. □

We now turn to define our online algorithm, HA (Hybrid Algorithm):
HA first classifies the items according to the following types. Define r's type as the tuple $T = (i, c)$ such that $l(I(r)) \in (2^{i-1}, 2^i]$ and $I(r)^- \in ((c-1)\cdot 2^i, c \cdot 2^i]$, such that $1 \leq i \leq \log\mu$ and $c \in \mathbb{N}$. Note that under such definition, given a specific i there may only be two types of items alive at any moment in time (i.e., two different values of c).

We further define two types of bins - GN (general) and CD (classify-by-duration) bins.
Non-formally, HA acts as follows. HA first checks for an open CD-type bin that holds type T items. If such a bin exists, HA will pack r into one of these bins in a First-Fit manner (opening a new CD-type bin if needed). Otherwise, it checks if the overall load of all active items of type T (including the load of the currently handled item) is strictly greater than $\frac{1}{2\sqrt{i}}$. If so, it opens a new CD-type bin and packs r into it, otherwise it packs r into a GN-type bin in a First-Fit manner (opening a new GN-type bin if needed). We note that HA does not need to know μ in advance, but rather adapts as μ grows.

In Algorithm 1 we formally define the algorithm.

Upon arrival of request r **do**
  $T = (i, c) \leftarrow r$'s type.
  $d \leftarrow$ overall load of all active items of type T, including r.
  if *(exists an open CD-type bin that holds type T)* **then**
    Pack r in a first-fit manner over all CD-type bins that hold type T. If needed, open a new CD-type bin for r.
  else
    if $(d \leq \frac{1}{2\sqrt{i}})$ **then**
      Pack r in a first-fit manner over all GN-type bins. If needed, open a new GN-type bin for r.
    else
      Open a new CD-type bin for r.
    end
  end
Upon departure of request r **do**
  Remove r from its bin and close bin if needed.

Algorithm 1: Description of the Hybrid Algorithm

We first state the main theorem of the section.

Theorem 3.2. *For any sequence of items, σ, $HA(\sigma) = O(\sqrt{\log\mu}) \cdot OPT_R(\sigma)$.*

Before proving Theorem 3.2, we first give the following lemma.

Lemma 3.3. *Given an input, σ, let GN_t denote the number of GN-type bins HA has open at time t. Therefore,*

$$GN_t \leq 2 + 4\sqrt{\log \mu}.$$

Proof. Items of type $T = (i, c)$ add a load of at most $\frac{1}{2\sqrt{i}}$ to the GN-type bins. Since there can be at most 2 types of items active at any given moment, for any given $i \in \{1, \ldots, \log \mu\}$, the overall load in all the GN-type bins is at most,

$$\sum_{i=1}^{\log \mu} \frac{1}{\sqrt{i}}. \tag{1}$$

Since, $\int \frac{1}{\sqrt{x}} = 2\sqrt{x}$ and the fact that $\frac{1}{\sqrt{x}}$ is non-increasing for $x \geq 1$, we get,

$$\sum_{i=1}^{\log \mu} \frac{1}{\sqrt{i}} \leq 1 + \int_{1}^{\log \mu} \frac{1}{\sqrt{x}} \mathrm{d}x = 1 + 2\sqrt{\log \mu}. \tag{2}$$

By (1) and (2), the overall load in all the GN-type bins, at time t, is at most $1 + 2\sqrt{\log \mu}$. Furthermore, this bound holds for all t. Since $i \geq 1$, the load of each item that is packed into the GN-type bins, is at most $1/2$. Therefore, if HA would have opened more than $2 + 4\sqrt{\log \mu}$ GN-type bins, at the time of the opening of such a bin, we would have that the overall load of all items in all of the GN-type bins would be strictly greater than $1 + 2\sqrt{\log \mu}$ which would have lead us to a contradiction. Therefore, we know that $GN_t \leq 2 + 4\sqrt{\log \mu}$ as needed. □

Given a type of item, $T = (i, c)$, let k_t^T denote the number of open CD-type bins HA has for type T at time t. Furthermore, let $k_t = \sum_T k_t^T$. We would have liked to have shown that $\mathrm{OPT}_R^t(\sigma) \geq \max\{1, \frac{k_t}{4\sqrt{\log \mu}}\}$ which would have (as we will see) given us the desired competitive ratio. Unfortunately, this is not the case. However, we can make it correct by means of amortization. Specifically, by increasing the cost of OPT, using the following reduction.

Given an input σ, we convert it into an input σ' in the following way:
Let $r \in \sigma$, let $i \in \{1, \ldots, \log \mu\}$ be such that $l(I(r)) \in (2^{i-1}, 2^i]$ and let $c \in \mathbb{N}$ be such that $I(r)^- \in ((c-1) \cdot 2^i, c \cdot 2^i]$.
Given such r, define r' as, $I(r')^- = I(r)^-$ and $I(r')^+ = (c+1) \cdot 2^i$. Thus to convert σ to σ' we simply convert every item in σ as defined above.

Note that every two items of the same type (i.e., $T = (i, c)$) depart together once the reduction is applied. Further note that the reduction simply increases r's length by at most a multiplicative factor of 4. Therefore, we get the following two observations.

Observation 1. $span(\sigma') \leq 4 \cdot span(\sigma)$.

Observation 2. $d(\sigma') = \sum_{r' \in \sigma'} l(I(r')) \cdot s(r') \leq \sum_{r \in \sigma} 4 \cdot l(I(r)) \cdot s(r) = 4 \cdot d(\sigma)$.

The following corollary shows that the optimal algorithm does not lose too much due to the reduction.

Corollary 3.4. *For any sequence of items, σ, let σ' denote the items after the reduction is applied. If the items in σ form a continuous interval of active items, then,*

$$OPT_R(\sigma') \leq 16 \cdot OPT_R(\sigma).$$

Proof.

$$\begin{aligned} \mathrm{OPT}_R(\sigma') &\leq 2 \cdot span(\sigma') + 2 \cdot d(\sigma') \\ &\leq 8 \cdot span(\sigma) + 8 \cdot d(\sigma) \leq 16 \cdot \mathrm{OPT}_R(\sigma), \end{aligned}$$

where the first inequality is due to Lemma 3.1, the second is due to observations 1 and 2 and the last inequality is due to the *time−space* and *span* bounds. □

Recall that k_t is the number of any CD-type bins HA has open at time t.

Lemma 3.5. *Given a sequence of items, σ, let σ' denote the sequence after the above reduction is applied. Furthermore, let k_t be as defined above. Therefore,*

$$OPT_R^t(\sigma') \geq \max\{1, \frac{k_t}{4\sqrt{\log \mu}}\},$$

Proof. We first note that if $k_t = 0$ then the lemma immediately follows (since the items in σ form a continuous interval of active items and by the reduction, so do the items in σ'). Therefore, we shall assume the contrary from now on.

Consider all active items of type $T = (i, c)$ at time t, in σ. Let $\hat{r}_T$ be the first item to arrive out of all considered items. By the above reduction, since all items of a certain type depart together, we have that at the time of $\hat{r}_T$'s arrival, he was the only active item of that type. Furthermore, any items of that type that are released between the time of $\hat{r}_T$'s arrival and t, do not depart. Thus, we can give the following two observations.

Firstly, by the definition of HA, items of this type placed in the GN-type bins, plus the item that opened the first CD bin that accepts only items of this type (there exists such a bin since $k_t^T \geq 1$), contribute an overall load of at least $\frac{1}{2\sqrt{i}} \geq \frac{1}{2\sqrt{\log \mu}}$ (i being such that $T = (i, c)$). This is due to the fact that since HA opened such a bin, we know that exists a moment, $t' \leq t$, such that the overall load of active items of type T, in σ, is at least $\frac{1}{2\sqrt{i}}$ and since σ' is the result of the reduction applied to σ, we know that these items are active at time t as well (this is due to the fact that if items of the same type intersect, they must depart together).

Secondly, we will show that if $k_t^T \geq 2$, the other $k_t^T - 1$ bins of this type contribute an overall load of at least $\frac{k_t^T - 2}{2}$, in σ'. Due to the reduction, it is enough to show that if k CD-type bins for type T were opened until time t, then items packed into these bins, by HA, in σ, contribute a load of at least $\frac{k-1}{2}$ (this is due to that fact by the reduction's definition, all these items will be active at time t). We will prove this using induction on k.

For $k = 2$ the statement follows by the definition of first-fit algorithms. For $k > 2$, consider time t' when bin $k - 1$ was opened. By our induction hypothesis, we know that items of type T that have arrived before time t' contribute a load of $\frac{k-2}{2}$. Again, by the definition of first-fit algorithms, since bin k was opened, all items that have arrived after time t' and before t (including at time t) contribute a load of at least $1/2$, giving us the desired result.

Thus, due to the fact that these items depart together, the overall load contributed by type T items, in σ' is at least $\frac{1}{2\sqrt{\log\mu}} + \frac{k_t^T - 2}{2} \geq \frac{k_t^T}{4\sqrt{\log\mu}}$, if $k_t^T \geq 2$. Otherwise, $k_t^T = 1$, and the overall load is, once again, at least $\frac{1}{2\sqrt{\log\mu}} \geq \frac{k_t^T}{4\sqrt{\log\mu}}$.

If we sum over all types, by the time-space bound and since there is at least one active item at time t, we get,

$$\mathrm{OPT}_R^t(\sigma') \geq \max\{1, \frac{k_t}{4\sqrt{\log\mu}}\},$$

as needed. □

We now turn to prove our main theorem, i.e., Theorem 3.2.

Proof of Theorem 3.2. Let σ' be σ after we apply the reduction. Therefore,

$$\begin{aligned}
\mathrm{HA}_t(\sigma) &= \mathrm{GN}_t + k_t \leq 2 + 4\sqrt{\log\mu} + k_t \\
&\leq 2 + 2\cdot\max\{4\sqrt{\log\mu}, k_t\} \\
&= 2 + 8\sqrt{\log\mu}\cdot\max\{1, \frac{k_t}{4\sqrt{\log\mu}}\} \\
&\leq O(\sqrt{\log\mu})\cdot\mathrm{OPT}_R^t(\sigma'),
\end{aligned} \tag{1}$$

where the first inequality follows by Lemma 3.3 and the last by Lemma 3.5.

Putting it all together,

$$\begin{aligned}
\mathrm{HA}(\sigma) &= \int_t \mathrm{HA}_t(\sigma)\mathrm{d}t \leq O(\sqrt{\log\mu})\int_t \mathrm{OPT}_R^t(\sigma')\mathrm{d}t \\
&= O(\sqrt{\log\mu})\cdot\mathrm{OPT}_R(\sigma') \leq O(\sqrt{\log\mu})\cdot\mathrm{OPT}_R(\sigma) \\
&= O(\sqrt{\log\mu})\cdot\mathrm{OPT}_R(\sigma),
\end{aligned}$$

where the first inequality is due to (1) and the second inequality is due to Corollary 3.4. □

4 LOWER BOUND OF $\Omega(\sqrt{\mathrm{LOG}\,\mu})$

In this section we show a lower bound of $\Omega(\sqrt{\log\mu})$ with respect to the more general adversary, namely, an optimal algorithm that can never repack items. Recall that such an optimal algorithm is denoted as OPT_{NR}.

Definition 4.1. Define σ_t^* as the following sequence of items: At time t, release one item for each length in $\{1, 2, 4, \ldots, 2^{\log\mu}\}$ sequentially, from shortest to longest. Furthermore, let all items be of load $\frac{1}{\sqrt{\log\mu}}$.

Ren *et al.* [10] presented a 4-approximation (off-line) algorithm called the Dual Coloring Algorithm (note that this algorithm is not admitted repacking), that was described and analysed against an optimal algorithm that is admitted repacking. We restate their theorem using our notations.

Theorem 4.2. *For every sequence of items, σ, $DC(\sigma) \leq 4OPT_R(\sigma)$.*

Theorem 4.3. *For every on-line, deterministic and clairvoyant algorithm, ON, exists a sequence of items, σ, such that, $ON(\sigma) \geq \Omega(\sqrt{\log\mu})\cdot OPT_{NR}(\sigma)$.*

Note that this is the stronger case, in the sense that the lower bound holds even w.r.t. a non-repacking off-line algorithm.

Proof. Let $t_i = i$ for $i = 0, \ldots, \mu - 1$. We now turn to define our adversary.

For all $i = 0, \ldots, \mu-1$, release a prefix of $\sigma_{t_i}^*$ and stop as soon as ON opens $\sqrt{\log\mu}$ bins. Note that ON is indeed forced to open that many bins since $|\sigma_{t_i}^*| = \log\mu + 1$ and each items' load is $\frac{1}{\sqrt{\log\mu}}$, hence the overall load at that time is at least $\sqrt{\log\mu}$.

We now turn to analyse ON's competitive ratio compared to a repacking optimal off-line algorithm. Since at any moment, $t \leq \mu$, ON is forced to open $\sqrt{\log\mu}$ bins,

$$\mu\sqrt{\log\mu} \leq \mathrm{ON}(\sigma). \tag{1}$$

Let $S_t(\sigma)$ be the overall load of σ at time t. Furthermore, for any given moment t_i, let l_{t_i} denote the length of the last released item in $\sigma_{t_i}^*$ by the adversary (if no items were released, let $l_{t_i} = 0$).

By the definition of our adversary, the last released item in $\sigma_{t_i}^*$, for any t_i, forces ON to open a new bin for it. Therefore, ON must pay for the full duration of each one of these items. Thus, we get,

$$\sum_{i=0}^{\mu-1} l_{t_i} \leq \mathrm{ON}(\sigma). \tag{2}$$

Since the lengths of the items given at any moment, t_i, form a geometric series, the sum of their lengths is at most $2\cdot l_{t_i}$. Therefore,

$$\int_{t\in[0,2\mu]} S_t(\sigma) = d(\sigma) \leq \frac{1}{\sqrt{\log\mu}}\cdot 2\cdot\sum_{i=0}^{\mu-1} l_{t_i}. \tag{3}$$

Therefore,

$$\begin{aligned}
\mathrm{OPT}_R(\sigma) &\leq 2\cdot\int_{t\in[0,2\mu]}\lceil S_t(\sigma)\rceil \leq 4\mu + 2\cdot\int_{t\in[0,2\mu]} S_t(\sigma) \\
&\leq 4\mu + \frac{1}{\sqrt{\log\mu}}\cdot 4\cdot\sum_{i=0}^{\mu} l_{t_i} \leq \frac{8}{\sqrt{\log\mu}}\mathrm{ON}(\sigma),
\end{aligned} \tag{4}$$

where the first inequality follows from Lemma 3.1, the second inequality follows from (3) and the last inequality follows from (1) and (2).

Putting it all together,

$$\begin{aligned}
\mathrm{ON}(\sigma) &\geq \Omega(\sqrt{\log\mu})\cdot\mathrm{OPT}_R(\sigma) \\
&\geq \Omega(\sqrt{\log\mu})\cdot\mathrm{DC}(\sigma) \geq \Omega(\sqrt{\log\mu})\cdot\mathrm{OPT}_{NR}(\sigma),
\end{aligned}$$

where the first inequality follows from (4), the second inequality follows from Theorem 4.2 and the last inequality follows from the fact that DC is a non-repacking algorithm. □

5 UPPER BOUND OF $O(\mathrm{LOG\,LOG}\,\mu)$ W.R.T ALIGNED INPUTS

Recall the definition of aligned inputs, i.e., that items of length $\in (2^{i-1}, 2^i]$ may only arrive at multiples of 2^i and that items that arrive at the same moment, t, arrive with an arbitrary order (meaning that every item must be handled before the next one arrives). Note that, in particular this means that any items that arrive strictly after $c \cdot 2^i$ and strictly before $(c+1) \cdot 2^i$ must depart by time $(c+1) \cdot 2^i$, for any $c, i \in \mathbb{N}$.

By the definition of binary inputs, we may use the notation t^+ to denote time t after all items arriving at t have arrived and t^- to denote time t after all items that depart at time t have departed and before any other item arrives.

Given such an input, σ, with value μ, we first partition it in the following way. Consider time $t_0 = 0$ and consider all items that arrive at that time. Let μ' denote the length of the longest item that arrived at that time and let $\mu_0 = 2^{\lceil \log \mu' \rceil}$. By the definition of aligned inputs, all items that arrive in the time interval $[t_0, t_0 + \mu_0]$ must also depart in that time interval (this is due to the fact that if an item departs after $t_0 + \mu_0$ then by the definition of aligned inputs, it must have arrived at time t_0. However, in this case that would contradict the definition of μ_0). Therefore, we may remove all items that arrive in this interval (and denote them as σ_0), from σ, and manage them as a separate input. We continue decomposing σ in the same way, resulting in mutually disjoint inputs, $\sigma_0, \sigma_1, \ldots$ (in the sense that no two items from different inputs, intersect). Under such a partition, given any σ_i, changing its starting time, t_i, to 0, results in the input remaining aligned. Therefore, from now on we will assume each σ_i starts at time 0. We note that every σ_i is defined such that an item of length $(\frac{\mu_i}{2}, \mu_i]$ arrives at time 0. In particular, this means that $\mu_i \leq 2\mu$ (where μ is defined by the original input, σ).

This partition can be done in an online manner, therefore we may apply a $f(\mu_i)$-competitive algorithm to each σ_i separately and achieve a $f(2\mu)$-competitive algorithm (where μ is defined with respect to the original input, σ) assuming f is a non-decreasing function. Thus, from now on we shall assume that our input is aligned, an item of length μ arrives at time 0, all items arrive and depart during the time interval, $[0, \mu]$ and μ is a power of 2, i.e., $\mu = 2^n$ for some $n \in \mathbb{N}$.

Before defining our algorithm, CDFF (Classify-by-Duration-First-Fit), we first give a few definitions. We first partition the possible lengths of items, i.e., $\bigcup_{i=0}^{\log \mu} (2^{i-1}, 2^i]$ (note that the interval, $(1/2, 1]$, may only include items of length 1, however we use the entire interval for convenience of notation). By the definition of aligned inputs, at any moment t, the longest item that may arrive is bounded from above (e.g., items of length 1 may arrive any moment, however items of length μ may only arrive at time 0). For any given moment, t, let $(2^{m_t - 1}, 2^{m_t}]$ denote the longest interval for which items of length in that interval may arrive. Note that m_t is defined by t and can therefore be computed in an online manner before any items arrive.

We now turn to define CDFF on aligned inputs. At time $t^- = 0^-$, open $\log \mu + 1$ bins, $b_0^1, \ldots, b_{\log \mu}^1$ (some of these bins may remain empty throughout the process). We note that by the definition of aligned inputs, items of any length may arrive. Given an item, r, CDFF first classifies it according to its length, i.e., $l(r) \in (2^{i-1}, 2^i]$, and then packs r into bin $b_{\log \mu - i}^1$. Note that items with larger intervals are packed into bins with smaller lower indexes. For example, an item of length μ is packed into b_0^1 and item of length 1 is packed into $b_{\log \mu}^1$. Once b_i^1 is too full to accept an item, we open b_i^2 and so on and so forth. Meaning, item of length $\in (2^{i-1}, 2^i]$ is put into bin $b_{\log \mu - i}^j$ for minimal j s.t. $b_{\log \mu - i}^j$ can accept said item. Later on, we will address the bins $\{b_i^j\}_j$, for a given i, as the i^{th} row of bins.

We note that although CDFF is defined as having prior knowledge of μ, at time $t^- = 0^-$ it packs items according to type and may therefore adapt as larger items arrive. This means it does not in fact need any prior knowledge of μ. Furthermore, items of length $\in (\frac{\mu}{2}, \mu]$ arrive at time 0, thus, during the following item arrivals CDFF will already know the value of μ.

We turn to define our algorithm for time $t^- > 1$:
Let $(2^{m_t - 1}, 2^{m_t}]$ denote the longest interval for which items of length in that interval may arrive. Now, CDFF first classifies items according to their length and then packs items that belong to the i^{th} smallest interval, into row $(m_t - i)$, in a first-fit manner (furthermore, once a bin b_i^j becomes empty, remove it from the i^{th} row and update indexes). Again we note that the bins' numbering is opposite to the interval numbering, i.e., items with interval i are packed into row $m_t - i$. Meaning that given an item r, if $l(r) \in (2^{i-1}, 2^i]$ for $0 \leq i \leq m_t$, then r is packed into $b_{m_t - i}^j$ for a minimal j that can accept it. Figure 1 shows a representation of what CDFF's bins will look like at any moment.

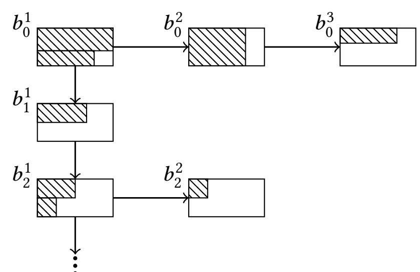

Figure 1: Representation of CDFF's bins at a given moment, t. Each rectangle represents a bin with some load.

In Algorithm 2 we formally define CDFF.

/* Denote b_i^j as bin j in row i of bins. */
Init: Open $log\mu + 1$ bins labelled $b_0^1, \ldots, b_{\log \mu}^1$.
Upon arrival of request r **do**
    $t^- \leftarrow r$'s arrival time.
    $(2^{m_t - 1}, 2^{m_t}] \leftarrow$ longest interval for which items of length in that interval may arrive.
    $i \leftarrow r$'s type, i.e., $l(I(r)) \in (2^{i-1}, 2^i]$.
    Pack r into row $m_t - i$ of open bins in a First-Fit manner. If needed open a new bin in that row.
Upon departure of request r **do**
    Remove r from its bin, close the bin and update indexes if necessary.

Algorithm 2: Description of CDFF Algorithm

We first state the main theorem of the section.

Theorem 5.1. *For any aligned input, σ,*

$$CDFF(\sigma) = O(\log\log\mu)OPT_R(\sigma).$$

Before proving our Theorem 5.1, in the next section we define a specific type of aligned input, namely, binary input, and show that for any binary input, σ_μ such that $\mu = 2^n$, we get $\text{CDFF}(\sigma_\mu) = O(\log\log\mu)\cdot\text{OPT}_R(\sigma_\mu)$. We then argue that in a sense, this is the worst case scenario and that in fact for any aligned input, σ, we have $\text{CDFF}(\sigma) = O(\log\log\mu)\cdot\text{OPT}_R(\sigma)$.

5.1 Upper Bound of $O(\log\log\mu)$ w.r.t Binary Inputs

Definition 5.2. Define a binary input of size $\mu = 2^n$, denoted as σ_μ, as the following collection of items:
For every $i \in \{0, 1, \ldots, \log\mu\}$, items of duration 2^i arrive at times, $0\cdot 2^i, 1\cdot 2^i, \ldots, (\frac{\mu}{2^i} - 1)\cdot 2^i$. Furthermore, all items' loads are $1/\log\mu$.

For example, figures 2 and 3, show what a binary input, σ_8 would look like and how CDFF would handle such an input.

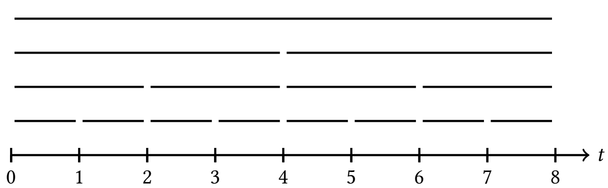

Figure 2: Representation of σ_8. Each segment represents an item.

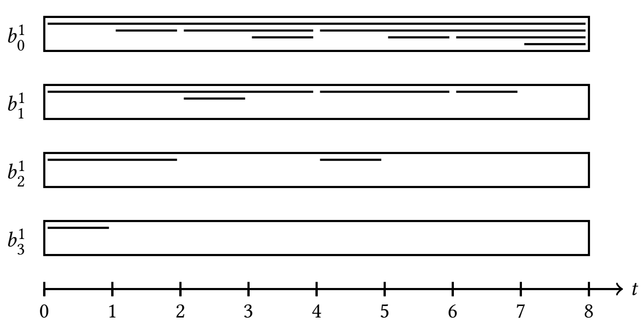

Figure 3: Representation of how CDFF packs σ_8.

Proposition 5.3. $\forall\mu$: *$CDFF(\sigma_\mu) \leq (2\log\log\mu + 1)OPT_R(\sigma_\mu)$.*

Surprisingly, our problem is closely related to the binary representation of time t and a few of its properties. In order to prove our proposition, we first give a few definitions and a few lemmas.

Definition 5.4. For $t \in \mathbb{N}$, let $binary(t)$ denote t's binary representation.

Since, by the definition of σ_μ, at time t only items of length 2^i such that $t/2^i \in \mathbb{N}$, may arrive, we get the following observation.

Observation 3. *Given a binary input, σ_μ, and for any $t \in \{0, 1, \ldots, \mu-2, \mu-1\}$, the number of requests that arrive at time t is equal to 1 + the length of the longest sequence of zeros that starts at the least significant bit (LSB) of $binary(t)$.*

For any $t \in \{0, 1, \ldots, \mu-1\}$ let $b_t = (1||binary(t)) \in \{0,1\}^{\log\mu+1}$ (where $a||b$ denotes the concatenation of binary strings, a and b). Furthermore, let $b_t = ((b_t)_{\log\mu}, \ldots, (b_t)_0)$.
Given σ_μ and time t, we define a mapping, f_t, from items in σ_μ to bits in b_t as follows.

$$\forall r \in \sigma_\mu : f_t(r) = (b_t)_{\log(l(I(r))}. \tag{1}$$

For example, if r is an item of length 1 given at time 0, $f_0(r) = (b_0)_{\log(1)} = (10\cdots0)_0 = 0$.

By the definition of σ_μ, an item of every length is active at every moment, and therefore this mapping is one-to-one and onto the $\log\mu$ least significant bits in b_t. Meaning that every item in σ_μ at any moment has an associated bit and every bit at every moment has an associated item.

Lemma 5.5. *Let σ_μ be a binary input and let $t \in \{0, 1, \ldots, \mu-1\}$. Therefore,*

1. *For any item $r \in \sigma_\mu$, if $f_t(r) = 1$ then r is assigned by CDFF to b_0^1.*
2. *For any item $r \in \sigma_\mu$, if $f_t(r) = 0$ and the maximal sequence of zeros that starts at $(b_t)_{\log(l(I(r))}$ and continues towards the most significant bit (MSB), is of size s (not including $(b_t)_{\log(l(I(r))}$), then r is assigned by CDFF to b_{s+1}^1.*

For example, if $b_t = 1001000$, then item of length 4 will be assigned to bin b_1^1.

Proof. We will prove this by induction on t.

For $t = 0$, $b_0 = 10\cdots0$ and by the definition of CDFF only the item of length 2^k for $k \in \{0, 1, \ldots, \log\mu\}$, is assigned to bin $b_{\log\mu-k}^1$. Furthermore, only one item arrives of each length, meaning that $b_{\log\mu-k}^1$ will indeed have room for the item. Thus 1 and 2 follow.

For $t > 0$, let $b_t = 1||\alpha||\beta$ such that $\beta = 10\cdots0 \in \{0,1\}^k$, $\alpha \in \{0,1\}^{\log\mu+1-k}$ and $k \in \{1, 2, \ldots, \log\mu\}$. By Observation 3 we know that the arriving items are of lengths $1, 2, 4, \ldots, 2^{k-1}$. By our inductive hypothesis and since items of lengths greater than 2^{k-1} have arrived before time t, we know that these items were assigned to bins correctly. Therefore, it is enough to show that any item of length $l_i = 2^i$ for $i \in \{0, 1, \ldots, k-1\}$ is assigned to bin b_{k-1-i}^1.

Every item in σ_μ has a load of $1/\log\mu$ and at any moment there are $\log\mu$ active items, therefore for the entirety of σ_μ, no bin of type b_i^1 will ever be full (and we will never have to open a bin of type b_i^2). By the definition of our algorithm, item of length l_i is assigned to the row of bins, b_{k-1-i}, and will therefore be assigned to bin b_{k-1-i}^1 as needed. □

Definition 5.6. For an aligned input, σ, let $\text{CDFF}_{t^+}(\sigma)$ be the amount of open bins in CDFF at time t^+.

Definition 5.7. For $b \in \{0,1\}^n$, let $max_0(b)$ be the length of the longest consecutive sequence of zeros in b.

Corollary 5.8. $\forall t \in \{0, 1, \ldots, \mu - 1\} : CDFF_{t^+}(\sigma_\mu) = \max_0(binary(t)) + 1$.

Proof. If $t = \mu - 1$ then $b_t = \overbrace{(11 \cdots 1)}^{\log \mu + 1 \text{ bits}}$ and by Lemma 5.5 all items are assigned to b_0^1. On the other hand, $\max_0(binary(t)) = \max_0 \overbrace{(11 \cdots 1)}^{\log \mu \text{ bits}} = 0$, implying the correctness of our lemma in this case.

If $t < \mu - 1$, by Lemma 5.5, the item assigned to b_s^1 for maximal s, is the item that has an associated bit 0 with a maximal number of consecutive zeros (counting towards the MSB). This is exactly the bit which is 0 and starts the longest sequence of consecutive 0's in $binary(t)$. Therefore, the length of the longest consecutive sequence of 0's starting at the items' bit (and counting towards MSB) is equal to $\max_0(binary(t)) - 1$ and by Lemma 5.5 this item is assigned to $b^1_{\max_0(binary(t))}$. Since the rows are indexed beginning at 0, $\text{CDFF}_{t^+}(\sigma_\mu) = \max_0(binary(t)) + 1$. □

Lemma 5.9. *Let $b = (b_1, \ldots, b_n) \in \{0, 1\}^n$ be n i.i.d. bits such that $Pr[b_i = 1] = 1/2$. Therefore,*

$$E[\max_0(b)] \leq 2 \log n.$$

Proof. By the union bound we have,

$$\begin{aligned}\Pr[\max_0(b) \geq 2 \log n] &\leq \sum_{i=1}^{n-2\log n} \Pr[(b_i, \ldots, b_{i+2\log n}) = (0, \ldots, 0)] \\ &\leq n \cdot \frac{1}{2^{2\log n}} = \frac{1}{n}.\end{aligned}$$

Therefore,

$$\begin{aligned}\text{E}[\max_0(b)] &= \sum_{i=1}^{2\log n - 1} i \cdot \Pr[\max_0(b) = i] + \sum_{i=2\log n}^{n} i \cdot \Pr[\max_0(b) = i] \\ &\leq 2 \log n - 1 + n \cdot \Pr[\max_0(b) \geq 2 \log n] \leq 2 \log n.\end{aligned}$$

□

Corollary 5.10. $\sum_{t=0}^{\mu-1} max_0(binary(t)) \leq 2\mu \cdot \log \log \mu$.

Proof. Let $n = \log \mu$. Therefore,

$$\begin{aligned}\frac{\sum_{t=0}^{\mu-1} \max_0(binary(t))}{\mu} &= \frac{\sum_{b \in \{0,1\}^n} \max_0(b)}{2^n} = \text{E}[\max_0(b)] \\ &\leq 2 \log n = 2 \log \log \mu.\end{aligned}$$

□

We now turn to prove Proposition 5.3.

Proposition 5.3. $\forall \mu : CDFF(\sigma_\mu) \leq (2 \log \log \mu + 1) OPT_R(\sigma_\mu)$.

Proof.

$$\begin{aligned}\text{CDFF}(\sigma_\mu) &= \sum_{t=0}^{\mu-1} \text{CDFF}_t(\sigma_\mu) = \sum_{t=0}^{\mu-1} (\max_0(binary(t)) + 1) \\ &= \mu + \sum_{t=0}^{\mu-1} \max_0(binary(t)) \leq \mu + 2 \cdot \mu \cdot \log \log \mu \\ &\leq (2 \log \log \mu + 1) \text{OPT}_R(\sigma_\mu),\end{aligned}$$

where the first inequality is by Corollary 5.10, the second equality is by Corollary 5.8 and the last inequality is due to the fact that $\text{OPT}_R(\sigma_\mu) \geq \mu$. □

5.2 Upper Bound of $O(\log \log \mu)$ w.r.t Aligned Inputs

In this section we show that CDFF is $O(\log \log \mu)$ - competitive w.r.t. arbitrary aligned inputs. Recall the reduction we used in section 3:

Given an input σ, define σ' as follows:
Let $r \in \sigma$, let $i \in \{1, \ldots, \log \mu\}$ be such that $l(I(r)) \in (2^{i-1}, 2^i]$ and let $c \in \mathbb{N}$ be such that $I(r)^- \in ((c - 1) \cdot 2^i, c \cdot 2^i]$.
Given such r, define r' as, $I(r')^- = I(r)^-$ and $I(r')^+ = (c + 1) \cdot 2^i$.
Note that in our case, i.e. aligned inputs, $I(r)$ is always equal to $c \cdot 2^i$. Therefore, the reduction simply increases the item's departure time to the next multiple of 2^i.

We now turn to prove that CDFF is $O(\log \log \mu)$ competitive as we did in section 3 - given an input, σ, we compare CDFF(σ) to $\text{OPT}_R(\sigma')$. Then, by Corollary 3.4, we get the desired result.
First we give the following definition.

Definition 5.11. Given σ and σ' such that σ is an arbitrary aligned input and σ' is the input we get by applying the reduction to σ, we define $d_r^{t^+}(\sigma')$ to be the overall load of all items that have ever been packed into row r by CDFF (in σ) and that are active at time t^+ (in σ'). That is, if $e \in \sigma$, then its load is added to $d_r^{t^+}(\sigma')$ only if e was packed by CDFF into row r of bins and $t^+ \in I(e')$ s.t. e' is e after we apply the reduction.

Lemma 5.12. *Given an arbitrary aligned input, σ, if at any moment t^+, CDFF has k open bins in row r, then $d_r^{t^+}(\sigma') \geq \frac{k-1}{2}$.*

Proof. Consider CDFF's r'th row of bins, $b_r^1, \ldots, b_r^k$ at time t^+. We will prove by induction on k that the overall load of all items that have ever been put into one of these bins by CDFF and that are active at time t^+, in σ', is at least $\frac{k-1}{2}$.

The case $k = 1$ is trivial. We now assume we have $k \geq 2$ open bins at time t^+, $b_r^1, \ldots, b_r^k$. By our induction hypothesis we may assume that the overall load of all items that were packed into $b_r^1, \ldots, b_r^{k-1}$ and are active at time t^+ in σ', is at least $\frac{k-2}{2}$. We now show that the overall load of all items that were packed into all k bins and that are active at time t^+ in σ', is at least $\frac{k-1}{2}$. Consider an item that is packed into bin b_r^k and that is active at time t^+ in σ. Denote this item as $\widetilde{r}$, its arrival time as $t_{\widetilde{r}}$ and its load as $s(\widetilde{r})$. If $s(\widetilde{r}) \geq 1/2$ then together with our induction hypothesis, $d_r^{t^+}(\sigma') \geq \frac{k-1}{2}$ as needed.

Otherwise, consider $\widetilde{r}$'s arrival time. Since CDFF packed items into row r, in a First-Fit manner, we know that at time $t_{\widetilde{r}}$ each bin, $b_r^1, \ldots, b_r^{k-1}$, has a load of at least 1/2 and therefore the overall load is at least $\frac{k-1}{2}$. We argue that all items that are active at time $t_{\widetilde{r}}$, in σ, in bins $b_r^1, \ldots, b_r^{k-1}$ are also active at time t^+ in σ', thus concluding our proof. To that end, consider an item in one of the $k - 1$ bins, $b_r^1, \ldots, b_r^{k-1}$, that is active at time $t_{\widetilde{r}}$ and denote it as $\hat{r}$ and its arrival time as $t_{\hat{r}}$. By the definition of CDFF we know that if items that arrive at the same time are packed into the same row of bins, then they must depart at the same time (in σ'). Therefore, if $t_{\widetilde{r}} = t_{\hat{r}}$ then, $\hat{r}$ and $\widetilde{r}$ depart together (in σ'). Since $\widetilde{r}$ is active at time t^+ in σ, it is also active at time t^+ in σ', meaning that $t_{\hat{r}}$ is also active at time t^+ in σ', as needed.

Otherwise, $t_{\hat{r}} < t_{\widetilde{r}}$. Since $\hat{r}$ arrived before $\widetilde{r}$ in σ, this is also the case in σ' (since arrival times are not altered). Furthermore, since both items are active at time $t_{\widetilde{r}}$ in σ, this is also the case in σ' (since items' lengths are only increased). Meaning that in σ', $\hat{r}$ arrived strictly before $\widetilde{r}$ and their time intervals intersected. By the definition of σ', $\hat{r}$ must therefore depart after $\widetilde{r}$. In particular this means that $\hat{r}$ is active at time t^+ in σ', as needed. Thus, in any case, we get that $d_r^{t^+}(\sigma') \geq \frac{k-1}{2}$ as needed. □

We now turn to prove our main theorem.

Proof of Theorem 5.1. Let $C_r^{t^+}$ denote CDFF's number of open bins in row r at time t^+. Therefore,

$$\begin{aligned}\mathrm{CDFF}(\sigma) &= \sum_{t=0}^{\mu-1} \mathrm{CDFF}_{t^+}(\sigma) = \sum_{t=0}^{\mu-1}\sum_{i=1}^{R_{t^+}} C_i^{t^+} \\ &\leq \sum_{t=0}^{\mu-1}\sum_{i=1}^{R_{t^+}} (2 \cdot d_r^{t^+}(\sigma') + 1) \qquad (1) \\ &= \sum_{t=0}^{\mu-1}\sum_{i=1}^{R_{t^+}} 1 + 2 \cdot \sum_{t=0}^{\mu-1}\sum_{i=1}^{R_{t^+}} d_r^{t^+}(\sigma'),\end{aligned}$$

where the first inequality follows by Lemma 5.12.

The sum $\sum_{t=0}^{\mu-1}\sum_{i=1}^{R_{t^+}}(1)$ represents the number of bins of type b_j^1 open at any given time. We have shown an upper bound for this number in Proposition 5.3, since in σ_μ at any given time items of all lengths available at that time arrive, meaning every bin of such type that could be opened is indeed opened. Therefore,

$$\sum_{t=0}^{\mu-1}\sum_{i=1}^{R_{t^+}} 1 \leq \mu \cdot (2 \log\log \mu + 1). \qquad (2)$$

For any r and t, $d_r^{t^+}(\sigma')$ accounts for loads that are disjoint and therefore no load is counted twice. Thus,

$$\sum_{t=0}^{\mu-1}\sum_{i=1}^{R_{t^+}} d_r^{t^+}(\sigma') \leq d(\sigma'). \qquad (3)$$

Putting it all together,

$$\begin{aligned}\mathrm{CDFF}(\sigma) &\leq \mu \cdot (2 \log\log \mu + 1) + 2 \cdot d(\sigma') \\ &\leq (3 + 2\log\log\mu) \cdot \mathrm{OPT}_R(\sigma') \\ &\leq (8 + 16 \log\log\mu) \cdot \mathrm{OPT}_R(\sigma) \\ &= O(\log\log\mu) \cdot \mathrm{OPT}_R(\sigma),\end{aligned}$$

where the first inequality is due to (1), (2) and (3), the second inequality is due to the *time* − *space* and *span* bounds (note that the *span* bound follows from our assumption that an item of length μ arrives at time 0) and the last inequality is due to Corollary 3.4. □

6 CONCLUSIONS AND OPEN PROBLEMS

We provide a $O(\sqrt{\log \mu})$-competitive algorithm and show a matching lower bound of $\Omega(\sqrt{\log \mu})$ on the competitiveness of any online algorithm, both bounds are with respect to general inputs. We also provide a $O(\log\log\mu)$-competitive algorithm with respect to aligned inputs. A natural open question would be to either show that with respect to aligned inputs our algorithm is optimal (by improving the lower bound) or improving the upper bound by showing a better performing algorithm. Another natural continuation of the research would be to inspect other interesting families of inputs.

REFERENCES

[1] János Balogh, József Békési, and Gábor Galambos. New lower bounds for certain classes of bin packing algorithms. *Theor. Comput. Sci.*, 440-441:1–13, 2012.
[2] Mihai Burcea. *Online dynamic bin packing.* PhD thesis, University of Liverpool, UK, 2014.
[3] Joseph Wun-Tat Chan, Prudence W. H. Wong, and Fencol C. C. Yung. On dynamic bin packing: An improved lower bound and resource augmentation analysis. *Algorithmica*, 53(2):172–206, 2009.
[4] Sandy Heydrich and Rob van Stee. Beating the harmonic lower bound for online bin packing. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 41:1–41:14, 2016.
[5] 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.
[6] Edward G. Coffman Jr., M. R. Garey, and David S. Johnson. Dynamic bin packing. *SIAM J. Comput.*, 12(2):227–258, 1983.
[7] Yusen Li, Xueyan Tang, and Wentong Cai. On dynamic bin packing for resource allocation in the cloud. In *26th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA '14, Prague, Czech Republic - June 23 - 25, 2014*, pages 2–11, 2014.
[8] Yusen Li, Xueyan Tang, and Wentong Cai. Play request dispatching for efficient virtual machine usage in cloud gaming. *IEEE Trans. Circuits Syst. Video Techn.*, 25(12):2052–2063, 2015.
[9] Yusen Li, Xueyan Tang, and Wentong Cai. Dynamic bin packing for on-demand cloud resource allocation. *IEEE Trans. Parallel Distrib. Syst.*, 27(1):157–170, 2016.
[10] Runtian Ren and Xueyan Tang. Clairvoyant dynamic bin packing for job scheduling with minimum server usage time. In *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2016, Asilomar State Beach/Pacific Grove, CA, USA, July 11-13, 2016*, pages 227–237, 2016.
[11] Steven S. Seiden. On the online bin packing problem. *J. ACM*, 49(5):640–671, 2002.
[12] Mordechai Shalom, Ariella Voloshin, Prudence W. H. Wong, Fencol C. C. Yung, and Shmuel Zaks. Online optimization of busy time on parallel machines. *Theor. Comput. Sci.*, 560:190–206, 2014.
[13] Xueyan Tang, Yusen Li, Runtian Ren, and Wentong Cai. On first fit bin packing for online cloud server allocation. In *2016 IEEE International Parallel and Distributed Processing Symposium, IPDPS 2016, Chicago, IL, USA, May 23-27, 2016*, pages 323–332, 2016.
[14] André van Vliet. An improved lower bound for on-line bin packing algorithms. *Inf. Process. Lett.*, 43(5):277–284, 1992.
[15] Prudence W. H. Wong, Fencol C. C. Yung, and Mihai Burcea. An 8/3 lower bound for online dynamic bin packing. In *Algorithms and Computation - 23rd International Symposium, ISAAC 2012, Taipei, Taiwan, December 19-21, 2012. Proceedings*, pages 44–53, 2012.