Online bin covering with advice

Boyar, Joan; Favrholdt, Lene M.; Kamali, Shahin; Larsen, Kim S.

Published in:
Algorithms and Data Structures

DOI:
10.1007/978-3-030-24766-9_17

Publication date:
2019

Document version:
Accepted manuscript

Citation for pulished version (APA):
Boyar, J., Favrholdt, L. M., Kamali, S., & Larsen, K. S. (2019). Online bin covering with advice. In Z. Friggstad, J-R. Sack, & M. R. Salavatipour (Eds.), *Algorithms and Data Structures: 16th International Symposium, WADS 2019, Proceedings* (pp. 225-238). Springer. Lecture Notes in Computer Science Vol. 11646
https://doi.org/10.1007/978-3-030-24766-9_17

Online Bin Covering with Advice*

Joan Boyar[1], Lene M. Favrholdt[1], Shahin Kamali[2], and Kim S. Larsen[1]

[1] University of Southern Denmark, Odense, Denmark
{joan,lenem,kslarsen}@imada.sdu.dk
[2] University of Manitoba, Winnipeg, Manitoba, Canada
shahin.kamali@umanitoba.ca

Abstract. The bin covering problem asks for covering a maximum number of bins with an online sequence of n items of different sizes in the range $(0, 1]$; a bin is said to be covered if it receives items of total size at least 1. We study this problem in the advice setting and provide tight bounds for the size of advice required to achieve optimal solutions. Moreover, we show that any algorithm with advice of size $o(\log \log n)$ has a competitive ratio of at most 0.5. In other words, advice of size $o(\log \log n)$ is useless for improving the competitive ratio of 0.5, attainable by an online algorithm without advice. This result highlights a difference between the bin covering and the bin packing problems in the advice model: for the bin packing problem, there are several algorithms with advice of constant size that outperform online algorithms without advice. Furthermore, we show that advice of size $O(\log \log n)$ is sufficient to achieve a competitive ratio that is arbitrarily close to $0.53\overline{3}$ and hence strictly better than the best ratio 0.5 attainable by purely online algorithms. The technicalities involved in introducing and analyzing this algorithm are quite different from the existing results for the bin packing problem and confirm the different nature of these two problems. Finally, we show that a linear number of bits of advice is necessary to achieve any competitive ratio better than 15/16 for the online bin covering problem.

1 Introduction

In the bin covering problem [3], the input is a multi-set of items of different sizes in the range $(0, 1]$ which need to be placed into a set of bins. A bin is said to be covered if the total size of items in it is at least 1. The goal of the bin covering problem is to place items into bins so that a maximum number of bins is covered. In the online setting, items form a sequence which is revealed in a piece-by-piece manner; that is, at each given time, one item of the sequence is revealed and an online algorithm has to place the item into a bin without any information about the forthcoming items. The decisions of the algorithm are irrevocable.

* The first, second, and fourth authors were supported in part by the Danish Council for Independent Research, Natural Sciences, grant DFF-1323-00247.

Bin covering is closely related to the classic bin packing problem and is sometimes called the dual bin packing problem[3]. The input to both problems is the same. In the bin packing problem, however, the goal is to place items into a minimum number of bins so that the total size of items in each bin is at most 1. Online algorithms for bin packing can naturally be extended to bin covering. For example, Next-Fit is a bin packing algorithm which keeps one "open" bin at any time: To place an incoming item x, if the size of x is smaller than the remaining capacity of the open bin, x is placed in the open bin; otherwise, the bin is closed (never used again) and a new bin is opened. Dual-Next-Fit [3] is a bin covering algorithm that behaves similarly, except that it closes the bin when the total size of items in it becomes at least 1.

In the offline setting, the bin packing and bin covering problems are NP-hard. There is an asymptotic fully polynomial-time approximation scheme (AFPTAS) for bin covering [17]. There are also bin packing algorithms which open $\textsc{Opt}(\sigma)+o(\textsc{Opt}(\sigma))$ bins [18,23,16], where $\textsc{Opt}(\sigma)$ is the number of bins in the optimal packing. The additive term was improved in [23] and further, to $O(\log \textsc{Opt}(\sigma))$, in [16].

Online algorithms are often compared under the framework of competitive analysis. An algorithm, $\mathbb{A}$, for bin covering (respectively, bin packing) is *c-competitive*, if there exists a constant b such that, for any input sequence, σ, $\mathbb{A}(\sigma) \geq c \cdot \textsc{Opt} - b$ (respectively, $\mathbb{A}(\sigma) \leq c \cdot \textsc{Opt} + b$). The *competitive ratio* of a bin covering (respectively, bin packing) algorithm, $\mathbb{A}$, is $\sup\{c \mid \mathbb{A} \text{ is } c\text{-competitive}\}$ (respectively, $\inf\{c \mid \mathbb{A} \text{ is } c\text{-competitive}\}$).

Despite similarities between bin covering and bin packing, the status of these problems are different in the online setting. In the case of bin covering, it is known that no online algorithm can achieve a competitive ratio better than 1/2 [13], while bin covering algorithms such as Dual-Next-Fit [3] have the best possible competitive ratio of 1/2. Hence, we have a clear picture of the complexity of deterministic bin covering under competitive analysis. The situation is more complicated for the bin packing problem. It is known that no deterministic algorithm can achieve a competitive ratio of 1.54278 [5] while the best existing deterministic algorithm has a competitive ratio of 1.5783 [4]. Note there is a gap between the best known upper and lower bounds.

Advice complexity is a formalized way of measuring how much knowledge of the future is required for an online algorithm to obtain a certain level of performance, as measured by the competitive ratio. When such advice is available, algorithms with advice could lead to semi-online algorithms. Unlike related approaches such as "lookahead" [15] (in which some forthcoming items are revealed to the algorithm) and "closed bin packing" [2] (where the length of the input is revealed), *any* information can be encoded and sent to the algorithm under the advice setting. This generality means that lower bound results under the advice model also imply strong lower bound results on semi-online algorithms, where

[3] There is another problem, also sometimes referred to as "dual bin packing", which asks for maximizing the number of items packed into a fixed number of bins; for the advice complexity of that dual bin packing problem, see [21,9].

one can infer impossibility results simply from the length of an encoding of the information a semi-online algorithm is provided with. Advice complexity is also closely related to randomization; complexity bounds from advice complexity can be transferred to the randomization case and vice versa [8,6,20,14].

The advice is generated by a benevolent oracle with unlimited computational power. The advice is written on a tape and the algorithm knows its meaning. This general approach has been studied for many problems (we refer the reader to a recent survey on advice complexity of online problems [11]). In particular, bin packing has been studied under the advice complexity [12,22,1].

Contributions

In this article, we provide the first results with respect to the advice complexity of the bin covering problem. To obtain an optimal result, advice essentially corresponding to an encoding of an entire optimal solution is necessary and sufficient. Not surprisingly, this follows from a similar proof for bin packing, since for both problems, bins filled to size one in an optimal solution are at the core of the proof. However, unlike the bin packing problem, advice of constant size cannot help improve the competitive ratio of algorithms. We establish this result by showing that any algorithm with advice of size $o(\log \log n)$ has a competitive ratio of at most 0.5, which is the competitive ratio of online algorithms without advice. We prove a tight result that advice of size $O(\log \log n)$ suffices to achieve a competitive ratio arbitrarily close to $0.53\bar{3}$. Finally, using a reduction from the binary string guessing problem [7], we show that advice of linear size is necessary to achieve any competitive ratio larger than 15/16. This is similar to, but more intricate, than the corresponding result for bin packing.

2 Optimal covering and advice

It is not hard to see that advice of size $O(n \log(\textsc{Opt}(\sigma)))$ is sufficient to achieve an optimal covering for an input σ of length n; note that $\textsc{Opt}(\sigma)$ denotes the number of bins in an optimal covering of σ. Provided with $O(\log(\textsc{Opt}(\sigma)))$ bits of advice for each item, the offline oracle can indicate in which bin the item is placed in the optimal packing. Provided with this advice, the online algorithm just needs to pack each item in the bin indicated by the advice. Clearly, the size of the advice is $O(n \log(\textsc{Opt}(\sigma)))$ and the outcome is an optimal packing. Note that it is always assumed that the oracle that generates the advice has unbounded computational power. However, if the time complexity of the oracle is a concern, we can use the AFPTAS of [17] to generate an almost-optimal packing and encode it in the advice. Similarly, if the input is assumed to have only m distinct known sizes, one can encode the entire request sequence, specifying for each distinct size how many of that size occur in the sequence. This only requires $O(m \log(n))$ bits of advice. The following theorem shows that the above naive solutions are asymptotically tight.

Theorem 1. *For online bin covering on sequences σ of length n, advice of size $\Theta(n \log \textsc{Opt}(\sigma))$ is required and sufficient to achieve an optimal solution, assuming $2\,\textsc{Opt}(\sigma) \leq (1-\varepsilon)n$ for some positive value of ε. When the input is formed by n items with $m \in o(n)$ distinct, known item sizes, advice of size $\Theta(m \log n)$ is required and sufficient to achieve an optimal solution.*

Proof. The lower bounds follow immediately from the corresponding results for bin packing [12, Theorems 1, 3]. Since the optimal result in those proofs have all bins filled to size 1, any non-optimal bin packing would also lead to a non-optimal bin covering. □

3 Advice of size $o(\log \log n)$ is not helpful

In this section, we show that advice of size $o(\log \log n)$ does not help for improving the competitive ratio of bin covering algorithms. This result is in contrast to bin packing where advice of constant size can improve the competitive ratio. Our lower bound sequence is similar to the one in [13], where the authors proved a lower bound on the competitive ratio of purely online algorithms.

Theorem 2. *There is no algorithm with advice of size $o(\log \log n)$ and competitive ratio better than $1/2$.*

Proof. Consider a family of sequences formed as follows:

$$\sigma_j = \langle \underbrace{\varepsilon, \varepsilon, \ldots \varepsilon}_{n \text{ items}}, \underbrace{1-j\varepsilon, 1-j\varepsilon, \ldots, 1-j\varepsilon}_{n/j \text{ items}} \rangle$$

Here, j takes a value between 1 and n and hence there are n sequences in the family. All sequences start with the same prefix of n items of size ε. We assume that $\varepsilon < \frac{1}{2n}$ to ensure that, even if all these items are placed in the same bin, the level of that bin is still less than $1/2$. Note that the suffix, formed by items of size $1 - j\varepsilon$ has length $O(n)$, and hence the length of all sequences is $\Theta(n)$.

Clearly, for packing σ_j, an optimal algorithm places j items of size ε in each bin and covers n/j bins. So we have $\textsc{Opt}(\sigma_j) = n/j$.

The proof is by contradiction, so assume there is an algorithm, $\mathbb{A}$, using $o(\log \log n)$ advice bits and having competitive ratio $1/2 + \mu$ for some constant $\mu > 0$. Thus, there exists a fixed constant d such that for any sequence σ_j we have

$$\mathbb{A}(\sigma_j) \geq (1/2 + \mu)\,\textsc{Opt}(\sigma_j) - d = \frac{n}{2j} + \frac{\mu n}{j} - d \tag{1}$$

We say two sequences belong to the same *sub-family* if they receive the same advice string. Since the advice has size $o(\log \log n)$, there are $o(\log n)$ sub-families. Let $\sigma_{a_1}, \ldots, \sigma_{a_w}$ be the sequences in one sub-family. Since the advice and the first n items (of size ε) are the same for any two members of this sub-family, $\mathbb{A}$ will place these n items identically. Let m_i denote the number of bins

receiving at least i items in such a placement. So, we have $\sum_{i=1}^{n} m_i = n$ (a bin with exactly x items is counted x times). Moreover, for any σ_j, we have

$$\mathbb{A}(\sigma_j) \leq m_j + (n/j - m_j)/2 = \frac{n}{2j} + \frac{m_j}{2} \tag{2}$$

This follows since any bin with at least j items of size ε can be covered using only one item of size $1 - j\varepsilon$, while the other bins require two such items.
From Equations 1 and 2, we get $\mu\frac{n}{j} \leq \frac{m_j}{2} + d$. Summing over $j \in \{a_1, \ldots, a_w\}$, we get that

$$\mu n \left(\frac{1}{a_1} + \frac{1}{a_2} + \ldots + \frac{1}{a_w}\right) \leq \frac{1}{2}(m_{a_1} + m_{a_2} + \ldots + m_{a_w}) + wd$$

Since $\frac{1}{2}(m_{a_1} + m_{a_2} + \ldots + m_{a_w}) + dw \leq \frac{1}{2} \cdot \sum_{i=1}^{n} m_i + dn = (d + \frac{1}{2})n$, we have

$$\frac{1}{a_1} + \frac{1}{a_2} + \ldots + \frac{1}{a_w} \in O(1)$$

Summing the left-hand side over all families, we include every sequence once and obtain $\Sigma_{i=1}^{n} \frac{1}{i}$. Since there are $o(\log n)$ sub-families, it follows that $\Sigma_{i=1}^{n} \frac{1}{i} \in o(\log n)$. This is a contradiction since the Harmonic number $\Sigma_{i=1}^{n} \frac{1}{i} \in \Theta(\log n)$.

Thus, our initial assumption is wrong and with advice of size $o(\log \log n)$, no algorithm with competitive ratio strictly better than $1/2$ can exist. □

4 An algorithm with advice of size $O(\log \log n)$

In this section, we show that advice of size $O(\log \log n)$ is sufficient to achieve a competitive ratio arbitrarily close to $0.53\bar{3}$. Throughout this section, we call an item *small* if it has size less than $1/2$ and *large* otherwise.

Consider a packing of the input sequence σ. We partition the bins in this packing into three groups. A *large-small (LS)* bin includes one large item and some small items, a *large-large (LL)* bin includes only two large items, and a *small (S)* bin includes only small items. We assume there is no small item in the LL bins of Opt (such small items can be moved to another bin without decreasing the number of covered bins). We also assume that, in Opt's packing, large items in LS bins are larger than those in LL bins (otherwise, we can move them around and LL bins will be still covered while the level of LS bins will increase). We use m and m', respectively, to denote the number of LS and LL bins in the optimal packing. For $m \geq 1$, we let $\beta \geq 1$ satisfy $m + m' = \beta m$. See Table 1 for a summary of notation used in this section.

In the following lemma and later, we use the algorithm Dual-Worst-Fit, which, given a fixed number of bins, places an item in a least full bin.

Lemma 1. *Given an integer q, assume we apply Dual-Worst-Fit to cover q bins. Let S denote the total size of packed items and d denote the maximum size of any item in the sequence. The level of any bin is at least $S/q - d$.*

Proof. The level of any two bins cannot differ by more than d; otherwise the last item placed in the bin with the larger level had to be placed in the bin with the smaller level. Let $B_{\min}$ and $B_{\max}$ be the two bins with minimum and maximum levels, respectively. From the above observation, we have $\text{level}(B_{\min}) \geq \text{level}(B_{\max}) - d$. The maximum level of any bin is no less than the average level of all bins. That is, $\text{level}(B_{\max}) \geq S/q$ which gives $\text{level}(B_{\min}) \geq S/q - d$. □

The following lemma shows that sequences with relatively few LS bins in an optimal packing are "easy" instances. The lemma is used in the case where $\beta \geq 15/14$, i.e., when there are at most 14 times as many LS bins as LL bins in the optimal packing. Note that, in this case, $(2\beta - 1)/(2\beta) \geq 8/15$.

Lemma 2. *There is an online bin covering algorithm with competitive ratio at least* $\min\{2/3, \frac{2\beta-1}{2\beta}\}$.

Proof. Consider a simple algorithm, $\mathbb{A}$, that places large and small items separately. Each pair of large items cover one bin and small items are placed using the Dual-Next-Fit strategy, that is, they are placed in the same bin until the bin is covered (and then a new bin is started). Let S denote the total size of small items. Note that the number of large items is $m+2m'$. The number of bins covered by $\mathbb{A}$ is at least $\lfloor (m + 2m')/2 \rfloor + \lfloor 2S/3 \rfloor$. The number of bins covered by OPT is at most $m + m' + \lfloor S \rfloor$. Thus, for any input sequence, σ,

$$\begin{aligned}\mathbb{A}(\sigma) &\geq \frac{\lfloor (m + 2m')/2 \rfloor + \lfloor 2S/3 \rfloor}{m + m' + \lfloor S \rfloor} \cdot \text{OPT}(\sigma) \\ &= \frac{(m + 2m')/2 + 2S/3}{m + m' + S} \cdot \text{OPT}(\sigma) - O(1).\end{aligned}$$

Table 1: Notation used in Section 4

Notation	Meaning
n	The length of the input
m	The number of LS bins in the optimal packing
m'	The number of LL bins in the optimal packing
S_l	An integer representing the total size of small items in the LS bins of the optimal packing (rounded down).
S_s	An integer representing the total size of small items in the S bins of the optimal packing (rounded down).
β	The value of $\frac{m+m'}{m}$. The algorithm behaves differently when $\beta \geq 15/14$ compared to when $\beta < 15/14$.
α	A parameter of the algorithm when $\beta < 15/14$. Approximately $\lfloor \alpha m \rfloor$ of covered bins include exactly one large item. We assume $\alpha < \frac{7-6\beta}{15} < \frac{4}{105}$.
k	An integer representing the precision of approximate encodings in $O(\log \log n)$ bits. We assume k is a large constant and we have $k \geq 6$.

This proves a competitive ratio of at least $\min\{2/3, \frac{2\beta-1}{2\beta}\}$, since

$$\frac{(m+2m')/2}{m+m'} = \frac{2m+2m'-m}{2m+2m'} = \frac{2\frac{m+m'}{m}-1}{2\frac{m+m'}{m}} = \frac{2\beta-1}{2\beta}.$$

□

Recall that among the large items, we assume that the largest m items form LS bins in the optimal packing. Let S_l and S_s be two integers that denote the floor of the total size of small items placed in respectively the LS and SS bins. So, the number of bins covered by OPT is at most $\beta m+S_s$. In what follows, we define (α, k)-desirable packings, which act as reference packings for our algorithm. Here α and k are two parameters of the algorithm that we will introduce later.

For the following definition, it may be helpful to confer with Figure 1.

Definition 1. *A covering is* (α, k)-desirable*, where α is a real number in the range $(0, 1]$ and k is a positive integer, if and only if all the following hold:*

- *I The covering has at least $\lfloor \alpha m \rfloor$ LS bins. All LS bins, except possibly a constant number of them, are covered.*
- *II The large items not in LS bins appear in pairs, with each pair covering one bin (except one item when there are an odd number of such large items).*
- *III The small items not in LS bins cover at least $\lfloor (1-\frac{1}{2^k})\frac{2S_s}{3} \rfloor - 1$ bins.*

Lemma 3. *For any input sequence, σ, the number of bins covered in an (α, k)-desirable packing is at least* $\min\{\frac{\alpha+2\beta-1}{2\beta}, (1-\frac{1}{2^k})\frac{2}{3}\} \cdot \text{OPT}(\sigma) - O(1)$.

Proof. The number of bins covered in the optimal packing is at most $m' + m + S_s = \beta m + S_s$. The number of bins covered by an (α, k)-desirable packing is at least $\lfloor \alpha m \rfloor - c$ (for covered LS bins; c is a constant) plus $\lfloor (2\beta - 1 - \alpha)m/2 \rfloor$ (for bins covered by pairs of large items) plus at least $\lfloor (1-\frac{1}{2^k})\frac{2S_s}{3} \rfloor - 1$ bins (covered by small items). So, the number, d, of bins covered in the (α, k)-desirable packing of σ will be

$$\begin{aligned} d &\geq \frac{\alpha m/2 + (2\beta-1)m/2 + (1-\frac{1}{2^k})2S_s/3}{\beta m + S_s} \cdot \text{OPT}(\sigma) - O(1) \\ &\geq \min\left\{\frac{\alpha+2\beta-1}{2\beta}, \left(1-\frac{1}{2^k}\right)\frac{2}{3}\right\} \cdot \text{OPT}(\sigma) - O(1). \end{aligned}$$

□

In the remainder of this section, we describe an algorithm that achieves an (α, k)-desirable covering for certain values of α and k. Here, k is used as a parameter to encode approximate values of a few numbers passed to the algorithm. Before describing these numbers, we explain how the approximate encodings work. Given a positive integer x, we can write the length of the binary encoding of x in $O(\log \log x)$ bits, using self-delimited encoding as in [19]. The approximate value of x will be be represented by the binary encoding of the length of x,

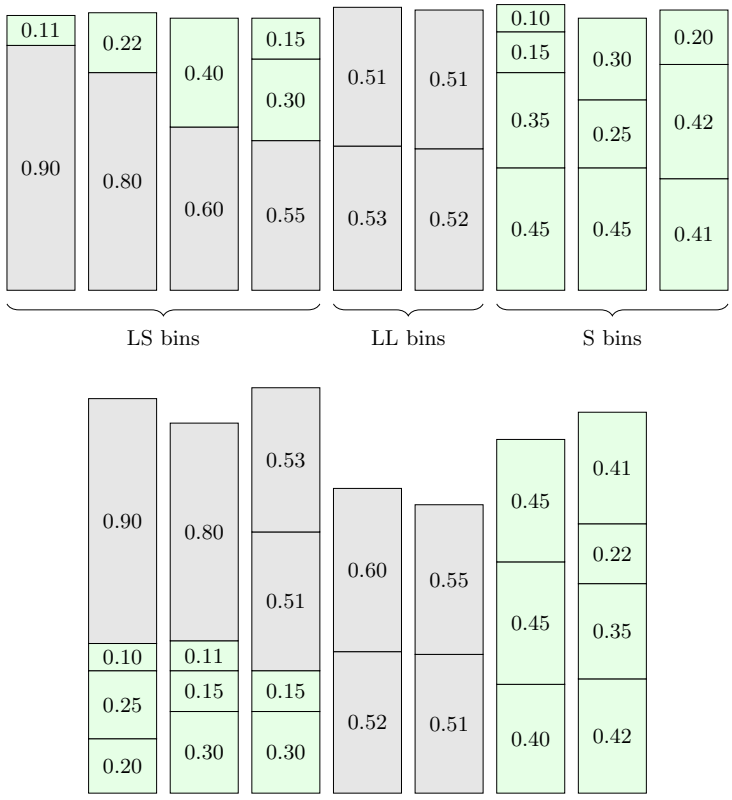

Fig. 1: (left) an optimal packing with $m = 4, m' = 2, S_l = \lfloor 1.18 \rfloor$ and $S_s = \lfloor 3.08 \rfloor$ (right) an (α, k)-desirable packing with $\alpha = 1/2$ and $k = 6$.

plus the k most significant bits of x after the high-order 1. Setting the unknown lower order bits to zero gives an approximation to x which we denote by $\bar{x}$. We can bound $\bar{x}$ as follows: If $\bar{x} = y \cdot 2^\ell$ for some y represented by $k+1$ bits, where the high-order bit is a one, then $2^k \leq y < 2^{k+1}$. Given $\bar{x}$, the largest x could be is $y \cdot 2^\ell + (2^\ell - 1)$. Thus, $(1 - \frac{1}{2^k})x < \bar{x} \leq x$.

In the remaining more technical part of the section, it may be beneficial to consider that if we had had $O(\log n)$ bits of advice instead of $O(\log \log n)$, many arguments would be simplified, and it could be helpful on a first reading to ignore multiplicative terms such as $1 - \frac{1}{2^k}$ that are there because we know only approximate as opposed to exact values of the parameters we receive information about in the advice.

First, we describe how the algorithm treats the small items and then discuss the large items. The algorithm receives $\bar{S_s}$ and $\bar{m}$, i.e., the approximate values of S_s and m, in $O(\log \log n + k)$ bits of advice. It places small items using the

Dual-Next-Fit strategy until a point at which the sum of small items observed so far becomes larger than $\bar{S}_s$. Let p be the small item that causes the sum to exceed $\bar{S}_s$. The algorithm places p and any other small item that follows it using the Dual-Worst-Fit strategy in $\lceil \bar{m}/3 \rceil$ bins, ignoring any large items when calculating the levels of the bins. In what follows, we refer to these $\lceil \bar{m}/3 \rceil$ bins as *reserved bins*. The items before p have a total size of more than $\bar{S}_s - 1$ and hence cover at least $\lfloor (2/3)(\bar{S}_s - 1) \rfloor \geq \lfloor 2\bar{S}_s/3 - 1 \rfloor \geq \lfloor (2/3)(1 - 1/2^k) S_s \rfloor - 1$. So, Property III of an (α, k)-desirable covering holds.

Next, we describe how the algorithm places large items so that properties I and II also hold. For that, the algorithm will need the approximate value of m (which was also required for small items) and m'. As before, these values can be encoded in $O(\log \log n + k)$ bits of advice. We call the largest $\lceil m/3 \rceil$ items in the input sequence *good items*. The algorithm aims at placing $\lfloor \alpha m \rfloor$ of the good items in the reserved bins. Before describing how the algorithm detects good items, we prove the following lemma, showing that the reserved bins with one good item will be covered.

Lemma 4. *A reserved bin that includes any good item will be covered in the final solution (covering) of the algorithm.*

Proof. Define the *desired level* to be $d = 1 - \text{size}(x)$ where x is the smallest good item. Consider an LS-bin B in the optimal packing that does not include a good item (that is, it has one large item smaller than any good item). The total size of small items in B will be at least d. As there are at least $\lceil 2m/3 \rceil$ such bins, we have that $S_l \geq (2m/3) \cdot d$, so $d \leq \frac{3S_l}{2m}$. On the other hand, the total size of small items placed in the reserved bins is at least $S_s + S_l - \bar{S}_s \geq S_l$. Since we use the Dual-Worst-Fit strategy to place these items into $m/3$ reserved bins, by Lemma 1, the total size of small items in any reserved bin is at least $\frac{S_l}{m/3} - y$ where y is the largest small item in those bins. Now, if a reserved bin includes a small item of size at least d, its level is already at least d; otherwise, $\frac{S_l}{m/3} - x$ will be at least $3S_l/m - d$ and since $d \leq \frac{3S_l}{2m}$, the level of the bins is at least d. □

So, in order to achieve an (α, k)-desirable packing, our algorithm needs to select $\lfloor \alpha m \rfloor$ good items and place them in the reserved bins; the above lemma indicates that these bins will be covered (Property I holds). Meanwhile, the algorithm ensures that other large items are paired and hence each pair of them covers a bin (Property II holds). In order to provide the above guarantees, the algorithm considers three cases depending on the location of good items (advice will be used to select the correct case).

Lemma 5. *When $\beta < 15/14$, there exists an (α, k)-desirable packing for $\alpha \leq \frac{7-6\beta}{15} - \frac{176-18\beta}{75 \cdot 2^k + 120}$ and k sufficiently large.*

Proof. Throughout the proof, $1 \leq \beta < 15/14$ and $\alpha < \frac{7-6\beta}{15} - \frac{176-18\beta}{75 \cdot 2^k + 120} < \frac{7-6\beta}{15} < \frac{1}{15}$. Note that under the description of the algorithm, we established Property III, and just prior to the statement of the lemma, we established Property I. The

proof to establish Property II is a case analysis on where good items appear in the request sequence.

Case 1: Assume there are $\lfloor \alpha\bar{m}/(1-1/2^k) \rfloor$ good items among the first $A = \lfloor \bar{m}/3 \rfloor$ large items in the sequence. In this case, the algorithm places the first A large items into the reserved bins. After seeing all these A items, the algorithm chooses the largest $\lfloor \alpha\bar{m}/(1-1/2^k) \rfloor \geq \lfloor \alpha m \rfloor$ of them and declares them to be good items, which by Lemma 4 are guaranteed to be covered. The remaining $A - \lfloor \alpha m/(1-1/2^k) \rfloor$ large items in the reserved bins will be paired with forthcoming large items. Since there are at least $m - A \geq 2m/3$ forthcoming large items and fewer than $m/3$ large items in the reserved bins waiting to be paired, all these large items (except possibly one) can be paired (Property II holds). In summary, in the final covering, there are $\lfloor \alpha\bar{m}/(1-1/2^k) \rfloor \geq \lfloor \alpha m \rfloor$ bins covered by a large item (and some small items) while the remaining large items are paired (except possibly one). Hence, the result will be an (α, k)-desirable packing.

Case 2: Assume there are fewer than $\lfloor \alpha\bar{m}/(1-1/2^k) \rfloor$ good items among the first A large items in the sequence (Case 1 does not apply). Furthermore, assume there are $\lfloor \alpha\bar{m}/(1-1/2^k) \rfloor$ good items among the $A = \lfloor \bar{m}/3 \rfloor$ large items that follow the first A large items. In this case, the algorithm places the first A large items pairwise in $\lceil A/2 \rceil$ bins. The A large items that follow are placed in the reserved bins. After placing the last of these items in the reserved bins, the algorithm considers these A items and declares the $\lfloor \alpha\bar{m}/(1-1/2^k) \rfloor \geq \lfloor \alpha m \rfloor$ largest to be good items, which by Lemma 4 are guaranteed to be covered. The remaining $A - \lfloor \alpha\bar{m} \rfloor$ reserved bins (with large items) will need to be covered by forthcoming large items. We know there are at least $m - 2A \geq \lfloor m/3 \rfloor$ forthcoming large items and fewer than $\lfloor m/3 \rfloor$ large items in reserved bins waiting to be paired, so all these large items (except possibly one) can be paired (Property II holds). Thus, the result is an (α, k)-desirable packing.

Case 3: Assume there are fewer than $\lfloor \alpha\bar{m}/(1-1/2^k) \rfloor$ good items among the first $A = \lfloor \bar{m}/3 \rfloor$ large items and also fewer than $\lfloor \alpha\bar{m}/(1-1/2^k) \rfloor$ good items among the following A items (Cases 1 and 2 do not apply). In what follows, we assume $\beta < 15/14$ and let α be some positive value such that $\alpha \leq \frac{7-6\beta}{15} - \frac{176-18\beta}{75 \cdot 2^k + 120}$. We will later choose k sufficiently large; here $k \geq 6$ will ensure that α is positive. Note also that we have $\alpha < \frac{7-6\beta}{15}$. In this case, the algorithm places the first $2A$ large items in pairs. There are $C = 2m' + m - 2A = 2m' + m - 2\lfloor \bar{m}/3 \rfloor$ remaining large items. The algorithm places the first $F = \bar{m}' + \lfloor \bar{m}/6 \rfloor + \lfloor \frac{\alpha\bar{m}}{2} \rfloor - 1$ of the last C large items in the reserved bins (note that this is roughly half of the last C large items when α is small). For this to be possible, we show that the number of reserved bins is at least F, and that at least $\alpha m - 6$ of the F items placed in the reserved bins are good items (see the full paper [10] for the calculations). After placing these F items, the algorithm declares the largest $\lfloor \alpha m \rfloor - 6$ among them to be good items. By Lemma 4, these items (along with small items in the reserved bins) will cover their respective bins. There are $F - \lfloor \alpha m \rfloor + 6$ (positive for $k \geq 1$ and $\alpha \leq \frac{1}{15}$) large items in reserved bins which have not been declared good, and the $C - F$ large items which have not arrived at this point will be paired with them. Calculations show that the number of large

items in the reserved bins which are not paired will be at most 6 (see the full paper [10]). The bins in which these items are placed, along with the $\lfloor \alpha m \rfloor - 6$ bins that include good items, will be the LS bins in the final (α, k)-desirable packing. Note that the number of these bins is $\lfloor \alpha m \rfloor$ minus an additive constant which is allowed in desirable packings. All large items placed in bins other than LS bins are paired and hence, Property II also holds. □

Theorem 3. *There is an algorithm that, provided with $O(\log \log n)$ bits of advice, achieves a competitive ratio of at least $\frac{12\beta-4}{15} - \frac{88-9\beta}{75\beta 2^k+120\beta}$, where k is a large but constant parameter of the algorithm. Since $\beta \geq 1$, for any $\varepsilon > 0$, there exists an algorithm using a sufficiently large k with competitive ratio at least $\frac{8}{15} - \varepsilon$.*

Proof. The advice indicates the values of $\bar{m}$, $\bar{m'}$, and $\bar{S}_s$. These values can all be encoded in $O(\log \log n)$ bits of advice. Note that one cannot calculate β exactly, since m and m' are not known exactly. Thus, the advice also includes 1 bit to indicate if Lemma 2 should be used because β is larger than 15/14. If not, the advice also indicates one of the three cases described above; this requires two more bits. Thus, the size of advice is $O(\log \log n)$.

If Lemma 2 is used, the competitive ratio is at least $\min\{2/3, \frac{2\beta-1}{2\beta}\}$, which for $\beta \geq 15/14$ is at least 8/15. Otherwise, provided with this advice and a sufficiently large integer parameter k, the algorithm can create an (α, k)-packing of the input sequence for any $\alpha \leq \frac{7-6\beta}{15} - \frac{176-18\beta}{75\cdot 2^k+120}$. By Lemma 3, the resulting packing has a competitive ratio of at least $\frac{\alpha+2\beta-1}{2\beta}$. Choosing $\alpha = \frac{7-6\beta}{15} - \frac{176-18\beta}{75\cdot 2^k+120}$ gives a scheme with competitive ratio at least $\frac{12\beta-4}{15\beta} - \frac{88-9\beta}{75\beta 2^k+120\beta}$. Since this is an increasing function of β and $\beta \geq 1$, the competitive ratio approaches $\frac{8}{15}$ for large values of k. □

5 Impossibility result for advice of sub-linear size

This section uses what is normally referred to as lower bound techniques, but since our ratios are smaller than 1, an upper bound is a negative result, and we refer to such results as negative or impossibility results. In what follows, we show that, in order to achieve any competitive ratio larger than 15/16, advice of linear size is necessary. We use a reduction from the binary separation problem:

Definition 2. *The* Binary Separation Problem *is the following online problem. The input $I = (n_1, \sigma = \langle y_1, y_2, \ldots, y_n \rangle)$ consists of $n = n_1 + n_2$ positive values which are revealed one by one. There is a fixed partitioning of the set of items into a subset of n_1* large *items and a subset of n_2* small *items, so that all large items are larger than all small items. Upon receiving an item y_i, an online algorithm must guess if y belongs to the set of small or large items. After the algorithm has made a guess, it is revealed to the algorithm which class y_i belongs to.*

A reduction from a closely related problem named "binary string guessing with known history" shows that, in order to guess more than half of the items correctly, advice of linear size is required:

Lemma 6. *[12] For any fixed $\beta > 0$, any deterministic algorithm for the Binary Separation Problem that is guaranteed to guess correctly on more than $(1/2+\beta)n$ input items on an input of length n needs at least $\Omega(n)$ bits of advice.*

The following lemma provides the actual reduction from the Binary Separation Problem to bin covering. The complete proof can be found in the full paper [10].

Lemma 7. *Consider the bin covering problem on sequences of length $2n$ for which* Opt *covers n bins. Assume that there is an online algorithm $\mathbb{A}$ that solves the problem on these instances using $b(n)$ bits of advice and covers at least $n - r(n)/8$ bins. Then there is also an algorithm* Bsa *that solves the Binary Separation Problem on sequences of length n using $b(n)$ bits of advice and guessing incorrectly at most $r(n)$ times.*

Proof. (sketch) Given an instance of the Binary Separation Problem formed by $n = n_1 + n_2$ values, we create an instance of the bin covering problem that starts with n_1 "huge" items of size $1 - \varepsilon$ for some $\varepsilon < \frac{1}{2n}$. Any "reasonable" bin covering algorithm has to place these items in separate bins. The next n items are created in an online manner, and each is associated with a value x in the Binary Separation Problem. The size of the item created for x will be an increasing function of x in the range $(\varepsilon, 2\varepsilon)$. We call the item associated with x "small" if x is small and "large" otherwise. If the bin covering algorithm places the item associated with x in a bin with a huge item, we guess that x is "small"; otherwise, we guess that x is "large". The last n_2 items of the bin covering instance are defined as complements of the large items. An optimal algorithm places small items in the bins opened for huge items and covers one bin with each large item and its complement. So, the number of covered bins in an optimal solution is n. We say an algorithm "makes a mistake" when it places a large item in a bin with a huge item or places a small item in a bin without a huge item. A detailed analysis shows that, for each 8 mistakes, the algorithm covers at least 1 bin fewer. Hence, if the number of covered bins is at least $n - r(n)/8$, then the number of binary separation errors must be at most $r(n)$. □

It turns out that reducing the Binary Separation Problem to bin covering (the above lemma) is more involved than a similar reduction to the bin packing problem [12]. The difference roots in the fact that there are more ways to place items into bins in the bin covering problem compared to bin packing; this is because many arrangements of items are not allowed in bin packing due to the capacity constraint.

Theorem 4. *Consider the bin covering problem on sequences of length n. To achieve a competitive ratio of $15/16 + \delta$, in which δ is a small, but fixed positive constant, an online algorithm needs to receive $\Omega(n)$ bits of advice.*

Proof. Suppose for the sake of contradiction that there is a bin covering algorithm $\mathbb{A}$ with competitive ratio $15/16 + \delta$ using $o(n)$ bits of advice. Consider sequences of length $2n$ for which Opt covers n bins. $\mathbb{A}$ covers $(15/16 + \delta)n =$

$n - r(n)/8$ bins for $r(n) = (1/2 - 8\delta)n$. Applying Lemma 7, we conclude that there is an algorithm that solves the Binary Separation Problem on sequences of length n using $o(n)$ bits of advice, while making at most $(1/2 - 8\delta)n$ errors. By Lemma 6, we know that such an algorithm requires $\Omega(n)$ bits of advice. So, our initial assumption that $\mathbb{A}$ required only $o(n)$ bits of advice is wrong. □

6 Concluding remarks

We have established that $\Theta(\log \log n)$ bits of advice are necessary and sufficient to improve the competitive ratio obtainable by purely online algorithms.

Obvious questions are: How much better than our bound of $8/15 = 0.53\overline{3}$ can one do with $O(\log \log n)$ bits of advice? Can one do better with $O(\log n)$ bits of advice?

References

1. Spyros Angelopoulos, Christoph Dürr, Shahin Kamali, Marc P. Renault, and Adi Rosén. Online bin packing with advice of small size. *Theory of Computing Systems*, 62(8):2006–2034, 2018.
2. Eyjólfur Ingi Ásgeirsson, Urtzi Ayesta, Edward G. Coffman Jr., J. Etra, Petar Momcilovic, David J. Phillips, V. Vokhshoori, Z. Wang, and J. Wolfe. Closed on-line bin packing. *Acta Cybernetica*, 15(3):361–367, 2002.
3. Susan F. Assmann, David S. Johnson, Daniel J. Kleitman, and Joseph Y.-T. Leung. On a dual version of the one-dimensional bin packing problem. *Journal of Algorithms*, 5(4):502–525, 1984.
4. János Balogh, József Békési, György Dósa, Leah Epstein, and Asaf Levin. A new and improved algorithm for online bin packing. In *26th Annual European Symposium on Algorithms (ESA)*, volume 112 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 5:1–5:14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018.
5. János Balogh, József Békési, György Dósa, Leah Epstein, and Asaf Levin. A new lower bound for classic online bin packing. *ArXiv*, 1807.05554 [cs:DS], 2018.
6. Hans-Joachim Böckenhauer, Juraj Hromkovic, and Dennis Komm. A technique to obtain hardness results for randomized online algorithms – A survey. In *Computing with New Resources – Essays Dedicated to Jozef Gruska on the Occasion of His 80th Birthday*, volume 8808 of *Lecture Notes in Computer Science*, pages 264–276. Springer, 2014.
7. Hans-Joachim Böckenhauer, Juraj Hromkovič, Dennis Komm, Sacha Krug, Jasmin Smula, and Andreas Sprock. The string guessing problem as a method to prove lower bounds on the advice complexity. *Theoretical Compututer Science*, 554:95–108, 2014.
8. Hans-Joachim Böckenhauer, Dennis Komm, Rastislav Královic, and Richard Královic. On the advice complexity of the k-server problem. *Journal of Computer and System Sciences*, 86:159–170, 2017.
9. Allan Borodin, Denis Pankratov, and Amirali Salehi-Abari. A simple PTAS for the dual bin packing problem and advice complexity of its online version. In *1st Symposium on Simplicity in Algorithms (SOSA)*, LIPIcs, pages 8:1–8:12. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

10. Joan Boyar, Lene M. Favrholdt, Shahin Kamali, and Kim S. Larsen. Online bin covering with advice. *ArXiv*, 1905.00066 [cs:DS], 2019.
11. Joan Boyar, Lene M. Favrholdt, Christian Kudahl, Kim S. Larsen, and Jesper W. Mikkelsen. Online algorithms with advice: A survey. *ACM Computing Surveys*, 50(2):19:1–19:34, 2017.
12. Joan Boyar, Shahin Kamali, Kim S. Larsen, and Alejandro López-Ortiz. Online bin packing with advice. *Algorithmica*, 74(1):507–527, 2016.
13. János Csirik and V. Totik. Online algorithms for a dual version of bin packing. *Discrete Applied Mathematics*, 21(2):163–167, 1988.
14. Christoph Dürr, Christian Konrad, and Marc P. Renault. On the power of advice and randomization for online bipartite matching. In *24th Annual European Symposium on Algorithms (ESA)*, volume 57 of *LIPIcs*, pages 37:1–37:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.
15. Edward F. Grove. Online bin packing with lookahead. In *6th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 430–436. SIAM, 1995.
16. Rebecca Hoberg and Thomas Rothvoss. A logarithmic additive integrality gap for bin packing. In *28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2616–2625. SIAM, 2017.
17. Klaus Jansen and Roberto Solis-Oba. An asymptotic fully polynomial time approximation scheme for bin covering. *Theoretical Computer Science*, 306(1–3):543–551, 2003.
18. Narendra Karmarkar and Richard M. Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *23rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 312–320. IEEE Computer Society, 1982.
19. Dennis Komm. *An Introduction to Online Computation – Determinism, Randomization, Advice*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2016.
20. Jesper W. Mikkelsen. Randomization can be as helpful as a glimpse of the future in online computation. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 39:1–39:14. Springer, 2016.
21. Marc P. Renault. Online algorithms with advice for the dual bin packing problem. *Central European Journal of Operations Research*, 25(4):953–966, 2017.
22. Marc P. Renault, Adi Rosén, and Rob van Stee. Online algorithms with advice for bin packing and scheduling problems. *Theoretical Computer Science*, 600:155–170, 2015.
23. Thomas Rothvoss. Approximating bin packing within o(log OPT * log log OPT) bins. In *54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 20–29. IEEE Computer Society, 2013.