Noname manuscript No.
(will be inserted by the editor)

Online Bin Packing with Advice of Small Size

Spyros Angelopoulos · Christoph Dürr · Shahin Kamali · Marc P. Renault · Adi Rosén

Received: date / Accepted: date

Abstract In this paper, we study the advice complexity of the online bin packing problem. In this well-studied setting, the online algorithm is supplemented with some additional information concerning the input. We improve upon both known upper and lower bounds of online algorithms for this problem. On the positive side, we first provide a relatively simple algorithm that achieves a competitive ratio arbitrarily close to 1.5, using constant-size advice. Our result implies that 16 bits of advice suffice to obtain a competitive ratio better than any online algorithm without advice, thus improving the previously known bound of $O(\log(n))$ bits required to attain this performance. In addition, we introduce a more complex algorithm that still requires only constant-size advice, and has a competitive ratio arbitrarily close to 1.47012. This is the currently best performance of any online bin packing algorithm

Research supported in part by project ANR-11-BS02-0015 "New Techniques in Online Computation–NeTOC". A preliminary version of this paper appeared in the Proceedings of the 14th International Symposium on Algorithms and Data Structures (WADS), 2015 [2].

Spyros Angelopoulos
CNRS and Sorbonne Universités, UPMC Univ Paris 06, LIP6, Paris, France
E-mail: spyros.angelopoulos@upmc.fr

Christoph Dürr
CNRS and Sorbonne Universités, UPMC Univ Paris 06, LIP6, Paris, France
E-mail: christoph.durr@lip6.fr

Shahin Kamali ✉
Department of Computer Science, University of Manitoba, Winnipeg, MB, R3T 2N2, Canada.
E-mail: shahin.kamali@umanitoba.ca

Marc P. Renault
Computer Sciences Department, University of Wisconsin - Madison, WI, USA.
E-mail: mrenault@cs.wisc.edu

Adi Rosén
CNRS and Université Paris Diderot, Paris, France.
E-mail: adiro@liafa.univ-paris-diderot.fr

with sublinear advice. On the negative side, we extend a construction due to Boyar et al. [13] so as to show that no online algorithm with sub-linear advice can be 7/6-competitive, improving on the lower bound of 9/8 from Boyar et al.

Keywords Online bin packing · Competitive analysis · Advice complexity

1 Introduction

Bin packing is a fundamental optimization problem that has played an important role in the development of approximation and online algorithms. An instance of the problem is defined by a set of *items* of different sizes, and the objective is to place these items into a minimum number of bins. For convenience, it is often assumed that the bins have capacity 1 and items have sizes in the range $(0, 1]$. In the online setting, the input set is revealed in a sequential manner, and the online algorithm must make an irrevocable decision concerning the placement of an item without any knowledge about the forthcoming items. We follow the canonical framework of competitive analysis of online algorithms [10], in which the performance of an algorithm $\mathbb{A}$ is determined by its competitive ratio, namely the maximum ratio between the cost of $\mathbb{A}$ (i.e., the number of bins opened by $\mathbb{A}$) and that of an optimal offline algorithm Opt for the same sequence. For the bin packing problem in particular, as often in the literature of the problem, we are interested in the *asymptotic* competitive ratio which considers sequences for which the costs of $\mathbb{A}$ and Opt are arbitrarily large. For this reason, throughout this paper we refer to the asymptotic competitive ratio of the bin packing problem as simply the competitive ratio.

The bin packing problem has provided some of the first-known explicit online algorithms (e.g., [20,25,26]). NextFit is a simple algorithm that maintains at each step a single open bin. If an incoming item fits in the bin, it is placed there; otherwise, that bin is closed and a new bin is opened to accommodate the item. FirstFit orders bins by their opening time and places an incoming item into the first bin which has enough space (opening a new bin if required). BestFit works similarly, except that it places the item into the bin with minimum available capacity which still has enough space for the item. It is known that NextFit is 2-competitive, whereas FirstFit and BestFit are both 1.7-competitive [26]. The best known online algorithm has a competitive ratio of 1.5815 [24]. No online algorithm can have a competitive ratio better than 1.54037 [4], a result that holds for both deterministic and randomized algorithms.

Competitive analysis, due to its inherent comparison to the offline optimum, often leads to a more pessimistic performance evaluation of online algorithms than what observed in practice [10]. Different models have been proposed in order to address this issue, and one such approach is to allow the online algorithm additional power. For example, the algorithm may be allowed to repack some items [18,19]. Alternatively, it may have access to lookahead [22], and, finally, may know the length of the input sequence [3] or

the value of Opt [17]. The advice model is a generalization of these approaches in which *any* information can be passed to the algorithm in the form of *advice*. In this sense, we can think of the advice as generated by a benevolent offline oracle with access to the entire input; the online algorithm can exploit the advice so as to produce a better solution. In principle, there is a certain correlation between the number of advice bits and the quality of the resulting solution. For many problems, including bin packing, a large number of advice bits is required in order to achieve optimal solutions; however, this does not imply that one may not achieve efficient (albeit non-optimal) solutions with significantly smaller number of bits.

In this paper, we study the impact of small-size advice (typically constant size) in improving the competitive ratio of bin packing algorithms. While our interest in studying the advice complexity stems from theoretical considerations, we note that small-size advice can be beneficial in practical settings. Consider for instance the setting in which the advice is transmitted through a noisy channel; in this setting, even small errors in the advice string may have a detrimental effect on the performance of the algorithm. Ensuring that the advice is small (ideally, of constant size) implies that the online algorithm is more resilient to such errors (eg.,in contrast to an algorithm that uses linear-size advice). For instance, one may add redundancy in the advice string in order to render it less prone to noise errors; therefore, if the advice string is small, the communication overhead is negligible.

In our work we use the following well-established definition of the bin packing problem under the advice setting.

Definition 1 In the *online bin packing problem with advice*, the input is a sequence of items $\sigma = \langle x_1, \ldots, x_n \rangle$, with $0 < x_i \leq 1$. At time step t, an online algorithm must pack item x_t into a bin, and this decision is a function of $\Phi, x_1, \ldots, x_{t-1}$, where Φ is the content of the advice tape. An algorithm has advice complexity $s(n)$ if it accesses at most $s(n)$ bits of an advice tape Φ for any input of length n.

1.1 Previous work and our contribution

The online advice model was first introduced by Emek et al. [16] and by Böckenhauer et al. [9,8]. These models were inspired by the work of Dobrev et al. [15], wherein the focus was on the advice required for optimality and the model allowed the oracle to provide information without a cost by using an empty string as the advice for a request. In the model of Emek et al. [16] an online algorithm receives a fixed number of bits of advice with each input item. Note that this model does not allow advice of sublinear size. In the model of Böckenhauer et al. [9], the advice is written on a read-only tape prior to the algorithm's execution, and the algorithm can read advice bits from that tape at will. The advice complexity has established itself as a prolific sub-field of online computation, and many online problems have been studied under the setting of online computation with advice (e.g., metrical task

systems [16], k-server problem [16,8,29,23], paging [14,9], list update problem [12], bin packing problem [13,30,2,27], knapsack problem [6], job shop scheduling [9,28], reordering buffer management problem [1], and minimum spanning tree problem [5]). For a comprehensive survey on recent advances in online computation with advice, we refer the reader to the survey [11].

In this paper, we study online bin packing under the advice-on-tape model. In this setting, Boyar et al. [13] proved tight bounds on the size of advice required to be optimal and showed that advice of super-linear size is necessary in order to attain optimality. They also proved that with advice of linear size, i.e., $\Theta(n)$ bits for a sequence of length n, one can achieve a competitive ratio of $4/3 + \epsilon$. This result was improved by Renault et al. [30] who showed that a competitive ratio arbitrary close to 1 can be achieved with $\Theta(n)$ bits. A related question is how many bits of advice are sufficient in order to outperform all online algorithms. For the bin packing problem, no online algorithm can have a competitive ratio better than 1.54037 [4], and Boyar et al. [13] showed that advice of size $\Theta(\log n)$ is sufficient to achieve an algorithm with competitive ratio of 1.5. They also proved that no algorithm is better than 9/8-competitive with advice of sub-linear size.

In our work, we address the power of small-sized advice in online bin packing. This is motivated by settings in which one may have some very limited information about the input, e.g., whether or not the input has many items of size beyond a certain threshold or some related statistical information. On the positive side, we prove that $O(1)$ advice suffices to outperform all online algorithms. More precisely, we first show that with only 16 bits of advice, we can achieve a competitive ratio of 1.530 (Section 2). Following a more complex approach, we show that constant-size advice suffices to go beyond the barrier of 1.5-competitiveness; more precisely, we achieve a competitive ratio arbitrarily close to 1.47012 (Section 3). This is, to date, the best upper bound on the competitiveness of any online algorithm that uses advice of sublinear size. We thus demonstrate the significant impact of small-size advice on the performance of bin packing algorithms. It should be mentioned that the simple algorithm of Section 2 reaches $1.5 + \epsilon$ (for any constant $\epsilon > 0$) with fewer advice bits than the complicated algorithm of Section 3. Last, we give a lower-bound construction that builds on ideas of [13] and which shows that advice of size $\Omega(n)$ is required to achieve a competitive ratio better than 7/6, thus improving the previous lower bound of 9/8 (Section 4).

In terms of techniques, for the upper bound of Section 2, we use information indirectly related to the ratio of "big" to "small" items; we show that this limited amount of information suffices to bring us arbitrarily close to the performance of algorithms that use logarithmic number of bits. For the more complicated upper bound of Section 3, the proof is based on two components which may be of independent interest. First, we introduce an algorithm for the special case that all items are larger than 1/3. In this case, any bin can include at most two items and the problem becomes a variant of the *upright matching problem* [21]. Interestingly, even this relaxed version of bin packing is not completely understood under the advice model. Nevertheless, we show that with

only one bit of advice we can achieve a competitive ratio of at most 1.3904. To achieve this goal, we devise two matching algorithms that complement each other; the single bit of advice indicates the better of the two algorithms for a given sequence. The second ingredient of our proof considers optimal packings in which each bin has an "empty space" of constant size ϵ. For sequences with such optimal packings, we show that advice of constant size is sufficient to achieve a competitive ratio of $1 + \epsilon$, using techniques based on grouping and rounding. To complete the proof, we show how to create offline packings in which items larger than $1/3$ are packed separately from the rest; as a result, the two algorithms mentioned above can be applied separately to reproduce such packing in an online manner (with advice of constant size).

Concerning the lower bound (Section 4), we base our construction on that of [13], using a reduction from the *string guessing problem* and a better amortization (i.e., counting) scheme that leads to an improvement of the bound. We note that following the conference version of this work [2], Mikkelsen [27] further improved the lower bound to $4-2\sqrt{2}$, using the same reduction and counting approach as this work, but a more involved, weighted counting scheme.

1.2 Preliminaries

Throughout the paper, for a given algorithm $\mathbb{A}$, we denote by $P^{\mathbb{A}}(\sigma)$ its output on sequence σ, namely a packing of the items in σ into bins. We also denote by $\mathbb{A}(\sigma) = |P^{\mathbb{A}}(\sigma)|$ the cost of $\mathbb{A}$ on σ, i.e., the number of bins opened by $\mathbb{A}$ on σ. We typically denote by r_j the j-th request of σ, when the sequence is clear from context.

Similar to previous works related to bin packing, we distinguish items based on their sizes. An item is *large* if it is larger that $1/2$ and an item is *small* if it is in the range $(0, 1/2]$.

These classes of items are further sub-divided: An large item is *huge* if it is larger than $2/3$, and is *critical* if it is in the range $(1/2, 2/3]$. A small item is *mini* if it is in the range $(1/3, 1/2]$, and *tiny* if it is in the range $(0, 1/3]$.

The *level* of a bin signifies the total size of the items in the bin. We say that a set of bins is *dedicated* to packing a class C of items if the set only contains items of class C.

2 Constant-size advice outperforms all online algorithms

In this section, we present an online algorithm, called RedBlue, that achieves a competitive ratio of $1.5 + \epsilon$ and uses a constant number of bits of advice. This algorithm is based on the ReserveCritical algorithm of [13] that uses $\Theta(\log n)$ bits of advice to indicate the number of critical items (items with a size in the range $(1/2, 2/3]$) and has a competitive ratio of 1.5. The challenging part is to show that RedBlue is able to approximate the packing of ReserveCritical using $O(1)$ advice bits, and thus achieve a competitive ratio arbitrarily close to 1.5.

First, consider the online algorithm ReserveCritical [13] that works as follows. Using $\Theta(\log n)$ advice bits, the number of critical items in the request sequence is encoded in binary. With this information, the algorithm initially opens a bin for each critical item and reserves a space of size $2/3$ for such an item. These bins are called *critical* bins. Then ReserveCritical proceeds to process the request sequence. The algorithm packs items based on their class as follows.

Huge Item: The item is packed in a new bin.
Critical Item: The item is packed in the reserved space of a critical bin. (Note that there will be one critical item per critical bin.)
Mini Item: The item is packed into a bin containing already a mini item. Otherwise, it is packed in a new bin. (Note that, for each bin with a mini item, there will be two mini items except possibly the last bin opened.)
Tiny Item: Using FirstFit, the item is packed into the non-reserved space of the critical bins. If the item does not fit in the non-reserved space of the critical bins, the item is packed into bins dedicated to tiny items.

As shown in [13], ReserveCritical has a competitive ratio of 1.5. To familiarize the reader with the salient ideas we will use later in this section, we first provide a simpler proof than the one given in [13].

Lemma 1 *Algorithm* ReserveCritical *has a competitive ratio of* 1.5.

Proof We consider two cases for the final packing produced by ReserveCritical for some request sequence σ. Initially, we consider the case that the packing contains bins with only tiny items and, then, we consider the case that the packing does not have any bin with only tiny items.

In the first case, we assume there is a bin dedicated to packing tiny items, and we show that the level of all bins (except potentially a constant number of them) is at least $2/3$. The level of all bins with both huge and mini items is no less than $2/3$ (except possibly the last bin opened for small items which may only contain a single small item). For each critical bin, there is a critical item and the space filled by the tiny items is at least $1/6$ (except possibly the last critical bin). For the sake of contradiction, assume that the space filled by the tiny items is less than $1/6$ in at least two critical bins, b_i and b_j, $i < j$. The tiny items in b_j could have been packed in the non-reserved space of b_i. This contradicts the FirstFit packing of the tiny items. Therefore, the level of all the critical bins is more than $1/2 + 1/6 = 2/3$. Finally, the level of all bins dedicated to packing the tiny items, except possibly the last one, is more than $2/3$ since all items in these bins have sizes $1/3$ or smaller. To conclude, all bins, with the exception of three bins (namely, one bin that contains only small items, the last bin opened for tiny items, and the last bin with a critical item), have a level of at least $2/3$ which guarantees a competitive ratio of at most 1.5.

In the second case, we assume there is no bin dedicated to packing tiny items. Using a weighting argument, we show that the competitive ratio is at most 1.5. We assign a weight to items based on their sizes. Huge and critical

items have a weight of 1; small items have a weight of 1/2; and tiny items have a weight of 0. The total weight of items in any given bin, except possibly one bin (namely, the bin with a single small item) is at least 1. Hence, we have that $A(\sigma) \leq W(\sigma)$, where $W(\sigma)$ is the total weight of items in σ. Further, the weight of any bin in an offline packing is at most 1.5, which occurs when a critical and a small item are placed in the same bin. Therefore, we have $\textsc{Opt}(\sigma) > W(\sigma)/1.5$ which completes the proof. □

In what follows, we present an online algorithm, called the RedBlue algorithm, that uses constant advice which is based on the final packing of ReserveCritical for the given request sequence σ. First, we describe the advice for RedBlue. The algorithm receives an integer i, $0 \leq i < 2^k$ encoded in binary, using k advice bits, where k is a constant independent of the length of the sequence. The value of i is determined by the packing of the ReserveCritical algorithm for σ. Let X and Y denote the number of bins in the packing of ReserveCritical of σ that include a critical item, and the number of bins opened for the tiny items, respectively. The advice for RedBlue encodes an approximate value of $\frac{X}{X+Y}$, using k bits, by encoding the value of i such that

$$\beta \stackrel{\text{def}}{=} \frac{i}{2^k} \leq \frac{X}{X+Y} < \frac{i+1}{2^k} = \beta + \frac{1}{2^k} \ . \tag{1}$$

Note that the above definition implies that β lies in the range $[0, 1]$. We now describe the algorithm. RedBlue always places each huge item in a single bin, and places mini items in dedicated bins, with two such items per dedicated bin. Further, the algorithm maintains a (possibly empty) set of *red* bins and a (possibly empty) set of *blue* bins. The red bins are dedicated to packing tiny items and the blue bins have a reserved space of 2/3 for critical items. To pack a critical item, RedBlue packs it in the reserved space of a blue bin without a critical item. If such a bin does not exist, a new bin is opened and declared as a blue bin. Next, we explain how RedBlue packs tiny items. Let the i-th request, y_i, be a tiny item. The algorithm applies FirstFit to place y_i in either the unreserved space of a blue bin, or in a red bin. If the algorithm cannot place y_i in one of the existing bins, it opens a new bin to pack y_i. This new bin is declared as either a red bin or a blue bin. To define the color of the bin, we consider three (exhaustive) cases for β:

Case I: when $\beta > 1 - 1/2^{k/2}$, the bin opened by a tiny item is always declared to be blue. Intuitively, since ReserveCritical opens a small number of bins with tiny items, we can afford to ignore red bins and declare all bins to be blue. At the end, 'a small' fraction of these blue bins might not receive a critical item and only include tiny items.

Case II: when $\beta < 1/2^{k/2}$, any new bis opened by tiny items is declared to be red. Intuitively, since there is a 'small' number of critical items, and we can afford 'not filling' them with extra tiny items.

Case III: assume $1/2^{k/2} \leq \beta \leq 1 - 1/2^{k/2}$. Let B_i and R_i denote the number of blue and red bins immediately after y_i is packed, respectively, where

$R_0 = B_0 = 0$. The algorithm will then declare the new bin as a blue bin if $\frac{B_{i-1}+1}{B_{i-1}+R_{i-1}+1} \leq \beta$; otherwise, it will declare the new bin as red. Note that, from the definition of the algorithm, RedBlue guarantees the invariant $\frac{B_i}{B_i+R_i} \leq \beta$ holds for all $i \leq n$. Further, it follows that the number of blue bins in the final packing of RedBlue is equal to X, i.e, the number of critical items in the sequence. In other words, for this case, since β is a lower bound on the ratio $\frac{X}{X+Y}$, all the bins declared as blue eventually receive a critical item.

Next, we analyze the algorithm. For cases I and III, we show that the cost of RedBlue is almost equal to that of ReserveCritical and apply Lemma 1 to prove an upper bound for competitive ratio of RedBlue. For case II, we directly compare RedBlue with Opt by showing that the level of almost all bins in RedBlue packing is at least 2/3. Let B and R respectively denote the number of blue and red bins in the final packing of RedBlue. The following three lemmas concern the three cases discussed above, respectively.

Lemma 2 *When $\beta > 1 - 1/2^{k/2}$, for the number of blue and red bins in* RedBlue *packing we have $B \leq (1 + \frac{5}{2^{k/2}}) \cdot (X + Y) + 1$ and $R = 0$.*

Proof Since $\beta > 1 - 1/2^{k/2}$, bins opened by tiny items are always blue and there is no red bin, i.e., $R = 0$. Moreover, from (1), we have

$$\frac{Y}{X+Y} \leq 1 - \beta < \frac{1}{2^{k/2}} \Rightarrow Y < \frac{1}{2^{k/2}} \cdot (X + Y) \ . \tag{2}$$

The first X bins in B match precisely the first X bins in the packing of ReserveCritical, i.e., they include X critical items plus the same tiny items. Let B' denote the last $|B| - X$ bins of B and let $Y' = |B'|$. Then, the bins of B' only include tiny items (i.e., the reserved space is not occupied by a critical item), and the level of the bins of B', except possibly the last one, is at least 1/6. For the sake of contradiction, assume that there are two bins, b_i and b_j, $i < j$, in B' with a level less than 1/6. The tiny items of b_j could have been packed in the non-reserved space of b_i which contradicts the FirstFit packing of the tiny items. Since the tiny items placed in the B' are the same items as those placed in the Y bins dedicated to packing tiny items in the packing of ReserveCritical, we have $Y' \leq 6Y + 1$ (because the level of bins of B' is at least 1/6). We can write: $B = X + Y' \leq X + 6Y + 1 < \left(1 + \frac{5}{2^{k/2}}\right) \cdot (X + Y) + 1$, where the last inequality comes from (2). □

Lemma 3 *When $\beta < 1/2^{k/2}$, the average level of all red and blue bins (excluding at most two red bins) is at least $\frac{3}{4}\left(1 - \frac{1}{2^{k-1}}\right)$.*

Proof Since $\beta < 1/2^{k/2}$, by (1) we have

$$\frac{X}{X+Y} < \beta + \frac{1}{2^k} < \frac{1}{2^{k/2}} + \frac{1}{2^k} < \frac{1}{2^{k-1}} \ . \tag{3}$$

All red bins, except possibly two bins, have level more than 3/4. To see this, consider the first bin that has level below 3/4 in the final packing. Since

we use First-Fit strategy to pack these items, any item that was packed in any later bin has size more than 1/4, and at most 1/3, guaranteeing a level of more than 3/4, apart from possibly the very last bin. Moreover, each blue bin has level more than 1/2 since it includes exactly one critical item. Hence, the average level of red and blue bins is more than $\frac{3}{4}\frac{R}{B+R} + \frac{1}{2}\frac{B}{B+R} \geq \frac{3}{4}\frac{R}{B+R}$. The number of blue bins is equal to the number of critical items, i.e., $B = X$. The number of bins opened by tiny items in the packing of RedBlue is no less than that of ReserveCritical, i.e, $R \geq Y$ (otherwise, the number of bins in RedBlue will be less than ReserveCritical and hence the average level bins in RedBlue will be at least equal to the average level of bins in ReserveCritical which is 2/3; see Lemma 1). Sine $B = X$ and $R \geq Y$, we have $\frac{R}{B+R} \geq \frac{Y}{X+Y}$. So, the average level of blue and red is more than $\frac{3}{4}\frac{Y}{X+Y} = \frac{3}{4}(1 - \frac{X}{X+Y}) > \frac{3}{4}\left(1 - \frac{1}{2^{k-1}}\right)$, where the last inequality follows from (3). □

Lemma 4 *When $1/2^{k/2} \leq \beta \leq 1 - 1/2^{k/2}$, for the number of blue and red bins in* RedBlue *packing we have $B + R < (X + Y)\left(1 + \frac{2}{2^{k/2}-2}\right) + 2^{k/2}$.*

Proof From (1) we have $\frac{X}{X+Y} < \beta + 1/2^k$ which implies $X < \frac{\beta+1/2^k}{1-\beta-1/2^k}Y$. In the given range for β, we have $\beta(1-\beta)2^k - \beta > 2^{k/2-1} - 1$. Hence,

$$\frac{1-\beta}{\beta}X < \left(1 + \frac{1}{\beta(1-\beta)2^k - \beta}\right)Y < \left(1 + \frac{1}{2^{k/2-1} - 1}\right)Y \tag{4}$$

Let y_j be a tiny item for which RedBlue opens the very last red bin in its packing. From the definition of the algorithm, we have $\frac{B_j+1}{B_j+R_j} > \beta$. This implies that $R_j < \frac{(1-\beta)B_j+1}{\beta}$. Recall that R and B are the number of red and blue bins in the final packing of the algorithm. We have $R = R_j$ and $B_j \leq B = X$. For the given range of β, in the final packing, all blue bins receive a critical item. Hence, $R \leq \frac{1-\beta}{\beta}X + 1/\beta$. From the above inequality, we obtain

$$B + R \leq X + Y + \left(\frac{2}{2^{k/2} - 2}\right)Y + 2^{k/2} < (X + Y)\left(1 + \frac{2}{2^{k/2} - 2}\right) + 2^{k/2}$$

Theorem 1 *For any $k \geq 4$, there is an online algorithm for the bin packing problem with k bits of advice that has competitive ratio $1.5 + \frac{15}{2^{k/2+1}}$.*

Proof We show RedBlue with k bits of advice achieves the claimed competitive ratio. When $\beta < \frac{1}{2^{k/2}}$ (case II), by Lemma 3, the average level of all red and blue bins (excluding the last red bin) is at least $\frac{3}{4}\left(1 - \frac{1}{2^{k-1}}\right)$. Meanwhile, the level of all bins opened by huge items and mini items (except possibly the last bin with mini items) is 2/3. Thus, on average, all bins in the packing of RedBlue (excluding two bins) have a level strictly more than $\frac{2}{3}\left(1 - \frac{1}{2^{k-1}}\right)$. We conclude that, in this case, the competitive ratio of the algorithm is less than $\frac{1}{2/3\cdot(1-1/2^{k-1})} < 3/2 + \frac{3}{2^k-2}$.

When $\beta > 1-1/2^{k/2}$ (case I), by Lemma 2, we have $B+R \leq r_1 \cdot (X+Y)+c_1$ for $r_1 = (1+\frac{5}{2^{k/2}})$ and some constant c_1. Similarly, when $1-1/2^{k/2} \leq \beta \leq 1/2^{k/2}$ (case III), by Lemma 4, we have $B+R \leq r_2 \cdot (X+Y)+c_2$ where $r_2 = (1+\frac{2}{2^{k/2}-2})$ and c_2 is a constant. In both cases, the number of bins opened by RedBlue for huge and mini items is the same as ReserveCritical. Note that if H and M denote the number of huge and mini items, then the cost of ReserveCritical can be described as $H + \lceil M/2 \rceil + X + Y$. For the cost of RedBlue, we have

$$\begin{aligned}\textsc{RedBlue}(\sigma) &= H + \lceil M/2 \rceil + B + R \leq H + \lceil M/2 \rceil + \max\{r_1, r_2\}(X+Y) + c \\ &< \max\{r_1, r_2\}\,(H + \lceil M/2 \rceil + X + Y) + c \\ &= \max\{r_1, r_2\}\,\textsc{ReserveCritical}(\sigma) + c \\ &\leq 1.5 \max\{r_1, r_2\}\,\textsc{Opt}(\sigma) + c'.\end{aligned}$$

The last inequality follows from Lemma 1, and c and c' are constants. To conclude, the competitive ratio of RedBlue is at most $\max\left\{1.5+\frac{3}{2^k-2}, 1.5r_1, 1.5r_2\right\}$ $= 1.5 + \max\left\{\frac{3}{2^k-2}, \frac{15}{2^{k/2+1}}, \frac{3}{2^{k/2}-2}\right\}$ which is $1.5+\frac{15}{2^{k/2+1}}$ when $k \geq 4$. □

In particular, for $k = 16$ bits of advice, we achieve a competitive ratio smaller than 1.530, which is strictly better than any online algorithm without advice.

3 Beyond 1.5-competitiveness with $O(1)$ advice bits

In this section, we present an online algorithm, called DR+ that achieves a competitive ratio that is arbitrarily close to 1.47012 while using only a constant number of advice bits. This algorithm outperforms any randomized online algorithm, since no online algorithm (deterministic or randomized) can achieve a competitive ratio better than 1.54037 [4]. Moreover, DR+ is a significant improvement over the previously best-known algorithm with sublinear advice which achieved a competitive ratio of 1.5, using, however, logarithmic advice [13].

We begin with a high-level description of DR+. The algorithm is defined so as to produce one of two possible packings, P_1 and P_2, of the input. A single bit of advice indicates to DR+ which of the two packings is the better approximation of the optimal packing. We show that, for any sequence, one of the two packings achieves the desired competitive ratio. To produce either packing, DR+ relies on a rounding scheme, that defines a constant number of item types, for all (P_1) or for a subset (P_2) of the items, and describes an approximate description of their offline packing. Provided in the advice string is either the exact or an approximate number of bins of each possible bin pattern in the offline packing, where a bin pattern describes the items types in the bin of the offline packing. For some inputs, the packing P_1 that applies the rounding scheme to all the items does not achieve the desired competitive

ratio. In such a situation, we must deal with the items of size more than 1/3 differently, and this is the packing P_2, where only the items of size at most 1/3 are packed using this rounding scheme.

First, we introduce an algorithm AlmostBestFit (Abf), that is defined for sequences in which all items are relatively large, namely larger than 1/3. Then we define an algorithm DesirableRounding (DR) that can approximate in an online fashion a class of particular offline packings, that we call desirable, using the rounding scheme. The algorithms Abf and DR are used as subroutines in the algorithm DR+ that is defined to handle arbitrary sequences. Specifically, DR is used to produce P_1 and both DR and Abf are used to produce P_2.

3.1 Sequences with items larger than 1/3

For this section, we will assume that σ is such that all its items are larger than 1/3. This implies that in every solution, each bin contains at most two items. We will present an algorithm, which we call AlmostBestFit (Abf) which uses only 1 bit of advice, and for which we will show that it has a competitive ratio that will be sufficient for our purposes, when moving to general sequences. In particular, we will show that Abf is 1.3904-competitive (Theorem 2).

Algorithm Abf is defined using the following two online algorithms which are variants of BestFit, namely SmallBestFit (Sbf) and LargeBestFit (Lbf). Before serving the sequence σ, Abf reads a single bit of advice that indicates which of Sbf or Lbf produces the best solution for σ.

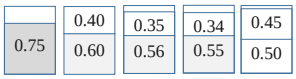
Sbf: All small items ($< 1/2$) are packed according to BestFit, and each large item ($\geq 1/2$) is placed in a new bin.

Lbf: All large items are packed according to BestFit, and each small item is placed in a new bin.

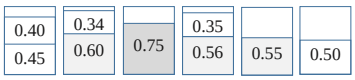
In order to show the upper bound on the competitive ratio of Abf, we require some notations that are defined based on a fixed optimal packing of σ, $P^{\text{Opt}}(\sigma)$. Let S be the set of small items, L be the set of large items packed in a bin with a small item in $P^{\text{Opt}}(\sigma)$ and H be the set of large items packed in a bin without a small item in $P^{\text{Opt}}(\sigma)$. Moreover, we can assume that all the large items in H are no smaller that the large items in L. Hence, the optimal cost is at least $\text{Opt}(\sigma) \geq |H| + |L|/2 + |S|/2$. Let $\text{Opt}_2(\sigma) = |L|/2 + |S|/2$.

In order to prove the competitive ratio of Abf, we use two parameters α and β such that $0 \leq \alpha \leq 1$ and $1 \leq \beta < 2$. The exact values of these parameters will be determined subsequently. In the optimal packing $P^{\text{Opt}}(\sigma)$, $|L|$ large items are matched with small items. We denote the pair of large and small items packed in a single bin as *partners*. Thus, the partner of a large item (respectively a small item) x is a small (respectively large) item which is placed in the same bin as x in $P^{\text{Opt}}(\sigma)$. Let $X \subseteq L$ be the set of large items which have their partners among the forthcoming items at the time they are packed (see Figure 1).

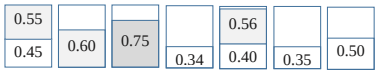
We consider the following three exhaustive cases:



(a) A Fixed Optimal packing



(b) Packing of SBF



(c) Packing of LBF

Fig. 1 Packings of Opt, Lbf, Abf for $\sigma = \langle 0.45, 0.60, 0.75, 0.34, 0.40, 0.56, 0.35, 0.55, 0.50\rangle$. We have $H = \{0.75\}$, $L = \{0.65, 0.56, 0.55\}$, and $S = \{0.40, 0.35, 0.34, 0.50, 0.45\}$. The set of large items which have their partners among forthcoming items is $X = \{0.60, 0.56\}$. For Lbf, the mapping of items in X to their partners change from $\{(0.60, 0.40), (0.56, 0.35)\}$ to $\{(0.60, 0.34), (0.56, 0.35)\}$ after placing 0.34. For Abf, the mapping changes from $\{(55, 34)\}$ to $\{(0.55, 0.45)\}$ after placing 0.55.

- Case I: $|L| \leq \frac{\beta-1}{2-\beta}|S|$;
- Case II: $|L| > \frac{\beta-1}{2-\beta}|S|$ and $|X| \geq \alpha|L|$; and
- Case III: $|L| > \frac{\beta-1}{2-\beta}|S|$ and $|X| < \alpha|L|$.

The observation that, in the final packing of Sbf, all small items (except potentially one of them) are placed with another item allows us to prove the following for Case I:

Lemma 5 *If* $|L| \leq \frac{\beta-1}{2-\beta}|S|$, Sbf *opens at most* $|H| + \beta \cdot \textsc{Opt}_2(\sigma) + 1$ *bins.*

Proof In the final packing of Sbf, all small items (except potentially one of them) are placed with another item. The worst case happens when all large items are single and all small items are placed together (two per bin). The cost of Sbf in this case is $|H| + |L| + |S|/2 + 1$. Note that the ratio $\frac{|L|+|S|/2}{|L|/2+|S|/2}$ is maximized when $L = \frac{\beta-1}{2-\beta}S$, from which we obtain that $\frac{|L|+|S|/2}{|L|/2+|S|/2} \leq \beta$. Hence $|L| + |S|/2 \leq \beta \cdot \textsc{Opt}_2(\sigma)$, and the lemma follows. □

Next, we consider Case II.

Lemma 6 *If* $|L| > \frac{\beta-1}{2-\beta}|S|$ *and* $|X| \geq \alpha|L|$, Sbf *opens at most* $|H| + (3/2 - \alpha/2) \cdot \textsc{Opt}_2(\sigma) + 1$ *bins.*

Proof We claim that, in the packing produced by Sbf for σ, at least $|X|$ small items are packed with large items. If this is true, then the lemma follows. More precisely, the number of bins opened by Sbf is at most $|H| + |L| + (|S| - |X|)/2 + 1 \leq |H| + |L| + (|S| - \alpha|L|)/2 + 1 = |H| + (2 - \alpha)|L|/2 + |S|/2 + 1$. Note that the ratio $\frac{(2-\alpha)|L|/2+|S|/2}{|L|/2+|S|/2}$ is maximized when $|L|$ as large as possible, namely when $|L| = |S|$. It follows that $\frac{(2-\alpha)|L|/2+|S|/2}{|L|/2+|S|/2} \leq \frac{3-\alpha}{2}$, from which we obtain that $(2 - \alpha)|L|/2 + |S|/2 \leq \frac{3-\alpha}{2}\textsc{Opt}_2(\sigma)$. We thus conclude that $\textsc{Sbf}(\sigma) \leq |H| + (3/2 - \alpha/2)\textsc{Opt}_2(\sigma) + 1$.

It remains to prove the claim that at least $|X|$ small items are packed with large items in the packing of SBF. To this end, we define a mapping of large items to small items that is of size $|X|$. We use $m(y)$ to denote the mapped item of an item y. A pair $(x, m(x))$ is called *valid* if it has the following properties:

(i) x is larger than $1/2$;
(ii) $x + m(x) \leq 1$; and
(iii) x appears earlier than $m(x)$ in the request sequence.

The mapping is said to be valid if all pairs in the mapping are valid.

Initially, the mapping is of the large items in X to their partner in $P^{\text{OPT}}(\sigma)$. Note that this is a valid mapping. We will show how to update this mapping, upon the arrival and packing of each item in σ by SBF, in such a way that, after packing all the items of σ, all $|X|$ pairs of mapped items are in the same bin in the packing produced by SBF.

Let y be the current item to pack. If y is larger than $1/2$, SBF opens a new bin for y and the mapping does not change.

Suppose that y is small and that the pair (z, y) is in the current mapping, for some large item z. If y is placed in the same bin as z, then the mapping does not change. If y is placed with another large item z', from the definition of $P^{\text{OPT}}(\sigma)$ and the fact that y is packed according to BESTFIT, we can assume that z' is no smaller than z. If, for any small item q, (z', q) is not in the mapping, we replace (z, y) with (z', y) in the mapping. Otherwise, $(z', m(z'))$ is in that mapping, and we replace (z, y) and $(z', m(z'))$ with (z', y) and $(z, m(z'))$, respectively. Note that $z + m(z') \leq z' + m(z') \leq 1$. In each case, the result is still a valid mapping.

Finally, suppose that y is small and it is not in the mapping. The mapping is not changed unless y is packed with a large item z such that (z, q) is in the mapping for some small item q. In this case, we replace the pair (z, q) with (z, y); this maintains a valid mapping. Note that at the time that y is packed, it cannot be packed with a small item q, where (z, q) is in the mapping for some large item z. Assume for the sake of contradiction that y is packed with q and that (z, q) is in the mapping. This requires q being placed alone in a bin at the time y appears. However, since (z, q) is in the mapping, when q is packed, z is alone in a bin (otherwise, the packing would have been updated to include $(z, m(z) \neq q)$. So, q cannot be alone in a bin at the time y is packed, contradicting z being placed with q.

□

Finally, it remains to consider Case III. The proof of the following lemma uses techniques similar to the proof of Lemma 6.

Lemma 7 *Suppose* $|L| > \frac{\beta-1}{2-\beta}|S|$ *and* $|X| < \alpha|L|$ *then the number of bins opened by* LBF *is at most* $|H| + (4 - 2(\alpha + \beta) + 2\alpha\beta) \cdot \text{OPT}_2(\sigma) + 1$.

Proof Let $Y = L \setminus X$ be the set of large items packed in a bin such that, for each $y \in Y$, the partner item of y occurs before y in σ. Note that $|Y| = |L| - |X|$. We claim that, in the packing produced by LBF for σ, at least

$|L| - |X| \geq (1-\alpha)|L|$ small items are packed with large items. If this is true, then the lemma follows. More precisely, the number of bins opened by LBF is at most $|H| + |L| + |S| - (1-\alpha)|L| + 1 = |H| + \alpha|L| + |S| + 1$. Note that $\frac{\alpha|L|+|S|}{|L|/2+|S|/2}$ is maximized for $|L|$ approaching $\frac{\beta-1}{2-\beta}|S|$ since $\frac{\alpha|L|+|S|}{|L|/2+|S|/2}$ is decreasing in L. It follows that $\frac{\alpha|L|+|S|}{|L|/2+|S|/2} \leq 4 - 2(\alpha+\beta) + 2\alpha\beta$. This implies that $\alpha|L| + |S| \leq (4 - 2(\alpha+\beta) + 2\alpha\beta) \cdot \text{OPT}_2(\sigma)$.

It remains to prove the claim that at least $|L|-|X| \geq (1-\alpha)|L|$ small items are packed with large items in the packing of LBF. This part is symmetric to the proof of Lemma 6 except that we consider the large items of Y instead of X. To this end, we once again define a mapping of large items to small items that is of size $|Y|$. We use $m(y)$ to denote the mapped item of an item y. A pair $(x, m(x))$ is called *valid* if it has the following properties:

(i) x is larger than $1/2$;
(ii) $x + m(x) \leq 1$; and
(iii) x appears later than $m(x)$ in the request sequence.

Note that this definition is the same as in Lemma 6 except for Property (iii). The mapping is said to be valid if all pairs in the mapping are valid.

Initially, the mapping is of the large items in Y to their partners in $P^{\text{OPT}}(\sigma)$. We will show how to update this mapping, upon the arrival and packing of each item in σ by LBF, in such a way, that after packing all the items of σ, all $|Y|$ pairs of mapped items are placed in the same bin by LBF.

Let y be the current item to pack. If y is smaller than $1/2$, LBF opens a new bin for y and the mapping does not change.

Suppose that y is large and that the pair (y, z) is in the current mapping, for some small item z. If y is placed in the same bin as z, then the mapping does not change. If y is placed with another small item z', from the definition of Y and the fact that y is packed according to BESTFIT, we can assume that z' is not smaller than z. If, for any large item q, (q, z') is not in the mapping, we replace (y, z) with (y, z') in the mapping. Otherwise, $(m(z'), z')$ is in the mapping, and we replace (y, z) and $(m(z'), z')$ with (y, z') and $(m(z'), z)$, respectively. Note that $z + m(z') \leq z' + m(z') \leq 1$. In each case, the result is still a valid mapping.

Finally, suppose that y is large and it is not in the mapping. The mapping is not changed after placing y unless y is packed with a small item z such that (q, z) is in the mapping for some large item q. In this case, we replace $(m(z), z)$ with (y, z); this maintains a valid mapping. □

The following theorem concludes the analysis of the ABF algorithm. The theorem will be used later in the proof of Lemma 14, in the context of general request sequences. Recall that α and β are parameter used for classifying the three exhaustive cases for which we proved Lemmas 5, 6, and 7. Assume we have $\alpha = (5 - \sqrt{17})/4$ and $\beta = (7 + \sqrt{17})/8$. For this selection of α and β we can state the following:

Theorem 2 *For a request sequence σ in which all items are strictly larger than 1/3,* ABF *opens at most $|H| + 1.3904 \cdot \text{OPT}_2(\sigma) + 1$ bins and uses one bit of advice.*

Proof From Lemmas 5, 6, and 7, SBF and LBF opens at most $|H|+\max\{\beta, 3/2-\alpha/2, 4-2(\alpha+\beta)+2\alpha\beta\} \cdot \text{OPT}_2(\sigma)+1$ bins, where $0 \leq \alpha \leq 1$ and $1 \leq \beta < 2$. The optimal choice for $\max\{\beta, 3/2-\alpha/2, 4-2(\alpha+\beta)+2\alpha\beta\}$ are the selected values of $\alpha = (5-\sqrt{17})/4$ and $\beta = (7+\sqrt{17})/8$ which gives a maximum value of at most $\beta < 1.3904$. □

The following corollary follows directly from Theorem 2, by observing that $\text{OPT}(\sigma) = H + \text{OPT}_2(\sigma)$.

Corollary 1 *For a request sequence σ in which all items are strictly larger than 1/3,* ABF *has a competitive ratio 1.3904 and uses one bit of advice.*

3.2 Arbitrary sequences

In this subsection, we define the algorithm DR+ and show that advice of constant size suffices to achieve a competitive ratio of $1.47012+\epsilon$ for any sequence and any arbitrarily small constant ϵ, $0 < \epsilon < 1/7$. The algorithm and its analysis build heavily on the ideas of grouping and rounding (discretization). We begin by defining the concepts of *ϵ-desirable* solutions and classes of *ϵ-desirable bins*.

Definition 2 A bin is *ϵ-desirable* and belongs to *class* 0 if there is an empty space of size more than ϵ in the bin.

A bin is *ϵ-desirable* and belongs to *class i* ($i \in \{1,2,3\}$) if its empty space is at most ϵ and if it contains exactly i items in the range $(1/i-\epsilon, 1/i]$.

An *ϵ-desirable packing* of a sequence σ is a packing formed by a set of *ϵ-desirable bins*.

We begin with an outline of our approach. Given $\epsilon \in (0, 1/7)$ and an optimal offline packing of a sequence σ, $P^{\text{OPT}}(\sigma)$, we will define two new packings P_1 and P_2. To create P_1 and P_2, we remove some items from bins and repack them together into new bins. P_1 and P_2 have the appealing properties that at least one of them provides a good approximation of the optimal packing and that both packings can be approximated in an online manner with constant advice. More precisely, P_1 is an ϵ-desirable packing of σ. The packing P_2 is comprised of two packings, P_{2a} and P_{2b}, of a partitioning of the items of σ. P_{2a} is a packing of the items with size at least 1/3, and P_{2b} is an ϵ-desirable packing of the items with size no more than 1/3. To approximate P_1, we use an algorithm DESIRABLEROUNDING (DR), which will be presented in Section 3.2.1. To approximate P_2, we use ABF (defined in Section 3.1) so as to approximate P_{2a} and DR so as to approximate P_{2b}. One additional bit of advice can thus determine the best among the two online approximations of P_1 and P_2.

3.2.1 Algorithm DesirableRounding

In this subsection, we will show that, for any packing P that consists of X ϵ-desirable bins, there is an online algorithm DesirableRounding (DR) which can pack the items of P, using $(1+\epsilon)X$ bins and requiring advice of size $f(\epsilon)$, where f is a function of ϵ. This is formalized in the statement of Lemma 8. The algorithm DR will be used as a subroutine in the complete algorithm defined in Section 3.2.2.

Lemma 8 *For a positive $\epsilon < 1$, let $P(\sigma)$ be an ϵ-desirable packing. Algorithm* DR *packs σ using at most $(1+\epsilon)|P(\sigma)|$ bins and an advice of size at most $(2\lceil\log_2(3/\epsilon+1)\rceil+1)\cdot(6\lceil 1/2\epsilon^2\rceil+1)e^{2\pi\sqrt{\lceil 1/2\epsilon^2\rceil}}$.*

First, we give an outline of the proof of Lemma 8. Given an ϵ-desirable packing $P(\sigma)$, the item sizes are rounded up so that there are m different item sizes or *item types*, where m is defined to be inversely proportional to ϵ^2. By applying this rounding scheme, there will be a constant (inversely proportional to ϵ^2) number of possible *bin patterns*, where the pattern of a bin is based on the number and types of the rounded items packed within. The advice provides for each bin pattern either the exact number of those bins in $P(\sigma)$, or an approximation of the fraction of those bins in $P(\sigma)$. Each of these values are encoded in $2k+1$ bits, where k is function of ϵ. Provided with this advice, DR opens bins of each pattern either with the same number as in $P(\sigma)$ or approximates this number by opening repeatedly bins in batches, maintaining proportions as given in the advice.

In what follows, we formalize the above intuition and provide a proof of Lemma 8. Let m be a positive integer that is a multiple of 6. A *bin pattern* is defined by a sequence of $m+1$ non-negative integers $(a_1,\ldots,a_m,c)$, specifying that it contains a_i slots of size i/m and one additional slot of size c/m, where $0 \le c \le m - \sum_{i=1}^{m} i a_i$. The last slot of size c/m is called the *tiny slot* if $c > 0$. The other slots are called *normal slots*. The following lemma bounds the number of bin patterns.

Lemma 9 *There are at most $(m+1)e^{\pi\sqrt{2m/3}}$ bin patterns.*

Proof A bin pattern can be viewed as a *partition* of the integer m, i.e., a way of obtaining m as a sum of integers, augmented with a possible mark for the tiny slot. It is known that the number of partitions is upper bounded by $e^{\pi\sqrt{2m/3}}$ (cf. [31]). The claim follows by multiplying the number of partitions by $m+1$ to account for the number of possible marks. □

Recall that ϵ-desirable bins of class $i \in \{1,2,3\}$ include i items in the range $(1/i-\epsilon, 1/i]$. We say an item is *ϵ-normal* if its size is no less than ϵ and *ϵ-tiny* otherwise.

With respect to the parameters m and ϵ, we define the notion of *nice bins* and *nice packings*. See Figure 2 for an illustration of this definition.

Definition 3 A bin b is said to be (m, ϵ)-*nice* for $\epsilon \geq 1/m$, if there exists a bin pattern and a mapping of the items of b to slots in the bin pattern with the following properties:

1. Every ϵ-normal item of size s is mapped to a normal slot of size $\lceil ms \rceil / m$ (which is s rounded up to the next multiple of $1/m$);
2. All ϵ-tiny items are mapped to a single tiny slot, which is big enough to contain them.

A packing is (m, ϵ)-nice if all the bins in the packing are (m, ϵ)-nice.

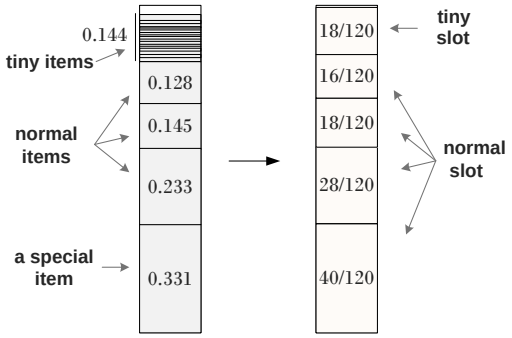


Fig. 2 An example of a (m, ϵ)-nice bin with $m = 120$ (left). If we increase item sizes to the closest value in $U = \{1/120, 2/120, \ldots, 120/120\}$, the total sizes of the resulting slots is not more than 1 (right).

We are now ready to define the algorithm DR. Let $P^{\text{OFF}}(\sigma)$ be an arbitrary (m, ϵ)-nice packing of σ by some (potentially offline) algorithm OFF. Based on $P^{\text{OFF}}(\sigma)$, we will first define the advice used by the algorithm and then its actions. At an intuitive level, the advice is used to encode, for each possible bin pattern, the number of bins in the packing of OFF. For a specific pattern, if the total number of bins can be encoded in $O(1)$ bits, then the exact number is encoded in the advice, otherwise an approximation of the ratio of the number of bins with the pattern over the total number of bins is encoded.

Advice definition for DR. Let $T(m)$ be the number of different bin patterns. For a bin pattern i $(1 \leq i \leq T(m))$, let η_i denote the number of bins with pattern i in $P^{\text{OFF}}(\sigma)$. It follows that $\sum_{i=1}^{T(m)} \eta_i = \text{OFF}(\sigma)$.

For each bin pattern i, define the bit

$$\alpha_i = \begin{cases} 1 & \text{if } \eta_i > 2^{2k} - 1 \\ 0 & \text{otherwise,} \end{cases}$$

and let $\beta_i := \left\lceil \frac{\eta_i}{\text{OFF}(\sigma)} \cdot (2^k - 1) \right\rceil$, where k is a positive integer whose value will be defined later. The value $\beta_i/(2^k - 1)$ approximates the fraction of bin pattern

in $P^{\text{OFF}}(\sigma)$. Hence, $\beta_i \in \{0, \ldots, 2^k - 1\}$ and the value of β_i can be encoded in k bits, whereas the value of η_i (when $\alpha_i = 0$) can be encoded in $2k$ bits.

For each bin pattern i, the advice for DR is the bit α_i followed by η_i if $\alpha_i = 0$, or β_i if $\alpha_i = 1$. The total size of advice is exactly $T(m) \cdot (1 + 2k)$.

Definition of DR. Algorithm DR maintains two sets of bins, denoted L_{DR} and L_{TINY}. The set L_{DR} is a set of bins with bin patterns described by the advice. The set L_{TINY} is a set of bins dedicated to packing ϵ-tiny items that overflow from L_{DR}. That is, the bins of L_{TINY} consist of a single tiny slot of size 1. Initially, for each bin pattern i where $\alpha_i = 0$, DR opens η_i bins and, for each bin pattern i where $\alpha_i = 1$, DR opens β_i bins. We designate an arbitrary tiny slot as active, assuming it exists. For each item $r_j \in \sigma$, DR packs it depending on its classification as an ϵ-normal, or ϵ-tiny item as follows.

Item r_j is ϵ-normal. Algorithm DR places r_j in a slot of size $\lceil m r_j \rceil / m$. If there is no empty slot of this size, then, for each bin pattern i where $\alpha_i = 1$, DR opens β_i new bins and packs r_j in a newly opened bin. The newly opened bins are added to L_{DR}.

Item r_j is ϵ-tiny. Algorithm DR places r_j in the active tiny slot. If there is no active tiny slot, Algorithm DR considers the cardinality of L_{TINY}.

- If $|L_{\text{TINY}}| < 2\epsilon \cdot |L_{\text{DR}}|$, then a new bin with a tiny slot of size 1 is added to L_{TINY}.
- Otherwise (i.e., $|L_{\text{TINY}}| \geq 2\epsilon \cdot |L_{\text{DR}}|$), for each bin pattern where $\alpha_i = 1$, DR opens β_i new bins and packs r_j in a newly opened bin. The newly opened bins are added to L_{DR}.

An arbitrary tiny slot among the newly opened is declared active. If the empty space in the active tiny slot drops below ϵ after placing r_j, the slot is closed, and, then an arbitrary non-closed tiny slot, if it exists, becomes the active tiny slot.

The following lemma shows that DR opens a number of bins no more that $(1 + 3\epsilon)$ the number of bins produced by any (m, ϵ) nice packing.

Lemma 10 *Let $\epsilon = 1/(2^k - 1)$ and consider a (m, ϵ)-nice packing of a sequence σ, denoted by $P^{\text{OFF}}(\sigma)$ produced by some algorithm* OFF. *The algorithm* DesirableRounding (DR) *opens at most $(1 + 3\epsilon)\,\text{OFF}(\sigma)$ bins and uses $(2\lceil \log_2(1/\epsilon + 1) \rceil + 1) \cdot (m + 1)e^{\pi\sqrt{2m/3}}$ bits of advice.*

Proof Denote by L^0_{DR} the initial set L_{DR} and by L^j_{DR} the set L_{DR} immediately after packing r_j. L^j_{TINY} is similarly defined.

From the definition of DR, the algorithm eventually opens $|L^n_{\text{DR}}| + |L^n_{\text{TINY}}|$ bins. We begin by bounding from above the number of bins in L^n_{TINY} and, then, L^n_{DR}.

First, from the definition of the algorithm, the number of L^n_{TINY} bins is at most $2\epsilon \cdot |L^n_{\text{DR}}|$. An important observation is that $L^n_{\text{DR}} \supseteq P^{\text{OFF}}(\sigma)$. Further, DR only opens a bin in L_{TINY} if all the tiny slots of L_{DR} have unused space less than ϵ. Since $L^n_{\text{DR}} \supseteq P^{\text{OFF}}(\sigma)$, we have that the volume of ϵ-tiny items

not packed in L^n_{DR} is at most $\epsilon P^{\mathrm{OFF}}(\sigma)$. For $\epsilon < 1/2$, these items will be packed into less than $(\epsilon/(1-\epsilon)) \cdot \mathrm{OFF}(\sigma) < 2\epsilon\,\mathrm{OFF}(\sigma)$ dedicated bins. Since $2\epsilon\,\mathrm{OFF}(\sigma) < 2\epsilon|L^n_{\mathrm{DR}}|$, we note that DR will open at most $\beta_i + \eta_i$ bins, for all i.

In order to show the correctness of the algorithm, we claim that whenever new bins are opened to pack an ϵ-tiny item then the new bins actually contain a tiny slot. We only need to consider the case $|L_{\mathrm{TINY}}| \geq 2\epsilon \cdot |L_{\mathrm{DR}}|$. Since there are ϵ-tiny items, there is at least one bin pattern i with a tiny slot in $P^{\mathrm{OFF}}(\sigma)$, and hence $\eta_i \geq 1$ and $\beta_i \geq 1$. If for one of those patterns i we have $\alpha_i = 1$ we are done. Otherwise all the tiny slots are in bin patterns with $\alpha_i = 0$, all of which are in L^0_{DR}. It follows from arguments above that the overflow can be packed in less than $2\epsilon \cdot |L^0_{\mathrm{DR}}|$ bins, which is a contradiction.

Last, we consider the set L^n_{DR}. As noted above, $L^n_{\mathrm{DR}} \supseteq P^{\mathrm{OFF}}(\sigma)$. Let $L^*_{\mathrm{DR}} = \min_i L^i_{\mathrm{DR}} \supseteq P^{\mathrm{OFF}}(\sigma)$. We claim that $|L^n_{\mathrm{DR}}| = |L^*_{\mathrm{DR}}|$, i.e. no more bins are added to L_{DR} after this point. As shown above, no additional bins will be opened when ϵ-tiny items are packed. As $L^*_{\mathrm{DR}} \supseteq P^{\mathrm{OFF}}(\sigma)$, there exists a slot in a bin of L^*_{DR} for each non-ϵ-tiny item in σ. Hence, DR will pack all the remaining non-ϵ-tiny items in L^*_{DR} without opening any additional bins. Therefore, for each bin pattern i where $\alpha_i = 0$, we have that the number of bins opened in DR is exactly η_i (as the advice is η_i and no additional bins are opened), and, when $\alpha_i = 1$, we have that the number of bins opened in DR is at most

$$\begin{aligned}
\eta_i + \beta_i &= (1 + \beta_i/\eta_i) \cdot \eta_i \\
&\leq \left(1 + \frac{2^k - 1}{2^{2k} - 1}\right) \cdot \eta_i \\
&\leq \left(1 + \frac{2^k - 1}{(2^k - 1)^2}\right) \cdot \eta_i \\
&= (1 + \epsilon)\eta_i \ ,
\end{aligned}$$

for $\epsilon = 1/(2^k - 1)$. Hence, $|L^n_{\mathrm{DR}}| \leq (1 + \epsilon)\,\mathrm{OFF}(\sigma)$.

Overall, DR opens $(1 + 3\epsilon)\,\mathrm{OFF}(\sigma)$ bins. By the definition of ϵ, we have $k = \lceil \log_2(1/\epsilon + 1) \rceil$. Using Lemma 9, the size of advice will be $(2k+1) \cdot T(m) = (2\lceil \log_2(1/\epsilon + 1) \rceil + 1) \cdot (m+1)e^{\pi\sqrt{2m/3}}$ which completes the proof.

□

We are now ready to prove Lemma 8.

Proof of Lemma 8. Without loss of generality we assume that ϵ is of the form $3/(2^k - 1)$ for some integer $k \geq 2$. We show that any ϵ-desirable packing is an (m, ϵ')-nice packing, where $\epsilon' = \epsilon/3 = 1/(2^k - 1)$ and $m = 6\lceil 1/(18\epsilon'^2) \rceil$. For the proof, we consider all the classes $\{1, 2, 3, 0\}$ of ϵ-desirable bins (see Definition 2).

Immediately from the definition, the bins of class $i \in \{1, 2, 3\}$ in $P(\sigma)$ are (m, ϵ')-nice. This is because the i items in the bins of class $i \in \{1, 2, 3\}$ can be

rounded up to $1/i$ within the capacity of the bin, even if it is larger than the next multiple of $1/m$.

Finally, we need to consider the bins of type 0 which have an empty space of size at least ϵ. In these bins, all ϵ'-normal items are rounded to the next multiple of $1/m$ to create a dedicated slot. The number of ϵ'-normal items is at most $1/\epsilon'$, hence the increase in volume by the rounding is at most $1/(\epsilon' m)$, which is less than ϵ by the choice of m and ϵ'.

In conclusion, $P(\sigma)$ is an (m, ϵ')-nice packing. Let Off be the algorithm that produced the packing $P(\sigma)$. By Lemma 10, DR opens at most $(1 + 3\epsilon')\,\textsc{Off}(\sigma) \leq (1+\epsilon)\,\textsc{Off}(\sigma)$ bins and uses advice of size $(2\lceil \log_2(1/\epsilon'+1)\rceil + 1)\cdot(m+1)e^{\pi\sqrt{2m/3}} = (2\lceil \log_2(3/\epsilon+1)\rceil+1)\cdot(6\lceil 1/(2\epsilon^2)\rceil+1)e^{2\pi\sqrt{\lceil 1/(2\epsilon^2)\rceil}}$. □

3.2.2 The complete algorithm DR+

The complete algorithm DR+ uses both Abf (Section 3.1) and DR (Section 3.2.1) as subroutines, depending on the optimal packing of the request sequence. Based on the optimal packing, two approximate packings are defined. In order to define these packings, we first need to distinguish between ϵ-*hard* and ϵ-*easy* bins as follows.

Definition 4 A bin is ϵ-*hard* if its two largest items x, y satisfy $x, y > 1/3$ and $x + y \geq 1 - \epsilon$. Otherwise, we call the bin ϵ-*easy*.

The following lemma implies that the items packed in a set of ϵ-easy bins can be packed into a set of ϵ-desirable bins without much overhead. This is accomplished by removing items from bins so as to make them ϵ-desirable. New bins are opened for these removed items in such a way that the items removed from 3 bins in the ϵ-easy packing account for 4 bins in the ϵ-desirable packing.

Lemma 11 *For $\epsilon < 1/7$, given a set of items packed in E ϵ-easy bins, it is possible to obtain an ϵ-desirable packing of these items using at most $4/3\cdot E+2$ bins.*

Proof The proof consists in two steps. In step 1, for each ϵ-easy bin that is not ϵ-desirable, we remove a subset of items of total size at most $1/3$ such that the resulting bin becomes ϵ-desirable. In step 2, we pack the removed items into ϵ-desirable bins.

Step 1: Let b be an ϵ-easy bin that is not ϵ-desirable. In particular, since it is not class 0 ϵ-desirable, its total size is at least $1 - \epsilon$.

1. If there exists an item y in b with size in $(\epsilon, 1/3]$, we remove y from b, making it class 0 ϵ-desirable.
2. Otherwise, if b includes any set S of items with total size in $(\epsilon, 2\epsilon]$, we remove S from b, making it class 0 ϵ-desirable. Note that such a set S contains a set of items of total size at least ϵ so that the size of each item is at most ϵ (note that if S contains an item larger than ϵ, we would be in case 1)

3. Otherwise the largest item x in b has size at least $1/3$. Let S be the remaining items in b.
   (a) If the size of S is at most ϵ, we remove S, making b a class 0 or class 1 ϵ-desirable bin.
   (b) Otherwise, the size of S is more than 2ϵ. Let $y \in S$ be the second largest item in b. The size of y cannot be less than or equal to ϵ, otherwise all other items in S are also smaller than or equal to ϵ and we would be in Case 2. Since we are not in case 1 and y is larger than ϵ, we conclude that y is bigger than $1/3$. Since b is not ϵ-hard, $x + y < 1 - \epsilon$. Let T be the set of all items in b except for the two largest items x and y. Since x and y are larger than $1/3$, the total size of items in T is at most $1/3$. Then we remove T, and b becomes a class 0 ϵ-desirable bin.

Step 2: Among the item sets removed in Step 1, we distinguish (i) singletons of size $(1/3 - \epsilon, 1/3]$ and (ii) other sets. The sets of class (i) are packed 3 by 3 into new class 0 or class 3 ϵ-desirable bins. The last opened bin may have less than 3 items.

For all other items sets (case ii), we proceed similarly except that we create class 0 ϵ-desirable bins, by the following argument. Since $\epsilon < 1/7$, we obtain that 3 item sets of size at most 2ϵ have total size less than $6\epsilon < 1 - \epsilon$. □

Similarly, we can convert ϵ-hard bins to ϵ-desirable bins.

Lemma 12 *For $\epsilon < 1/7$, given a set of items packed in H ϵ-hard bins, it is possible to obtain an ϵ-desirable packing of these items using at most $3/2 \cdot H + 2$ bins.*

Proof Define two class of ϵ-hard bins depending on whether the second largest item (i) is in the range $[1/2 - \epsilon, 1/2]$ or in the range $(1/3, 1/2 - \epsilon)$. For each pair of bins b_1 and b_2 from the same class C, we open a new bin and place the second largest items of b_1 and of b_2. As a result, the bins b_1 and in b_2 become class 0 ϵ-desirable bins. The new bin is either class 0 ϵ-desirable, or if its empty space is at most ϵ, then it contains two items in the range $[1/2 - \epsilon, 1/2]$ and hence is class 2 ϵ-desirable. Overall, for every two original ϵ-hard bins, three bins are created in the ϵ-desirable set of bins. The additive factor of 2 comes from the parity of the two classes of ϵ-hard bins in the original set.

The result in the above lemma is tight in the sense that there are instances of ϵ-hard bins so that converting them to any ϵ-desirable packing increases the number of bins by a factor of $3/2$. For example, consider a packing formed by H bins, each including two items of sizes $1/2 + \epsilon/2$ and $1/2 - \epsilon/2$. Any ϵ-desirable packing has H bins for large item of size $1/2 + \epsilon/2$ (without any other item), while any other bin can have at most two items of size $1/2 - \epsilon/2$. So, any ϵ-desirable packing has at least $3/2 \cdot H$ bins.

Fix an arbitrary packing $P^{\text{OFF}}(\sigma)$ and define H and E to be the number of ϵ-hard and ϵ-easy bins in $P^{\text{OFF}}(\sigma)$, respectively, and let γ denote the ratio H/E. In the following lemma, we define P_1 based on $P^{\text{OFF}}(\sigma)$ using Lemmas 11 and 12, and we approximate P_1 with DR.

Lemma 13 *For any request sequence σ and an $\epsilon < 1/10$, algorithm* DR *has a competitive ratio of $\frac{9\gamma+8}{6\gamma+6} + \epsilon$ and uses advice of size $O(2^{3.7(1+\gamma)/\epsilon} \cdot \log(\gamma/\epsilon))$.*

Proof We create an ϵ'-desirable packing P_1 with parameter $\epsilon' = \epsilon/(1+\gamma)$ from $P^{\text{OFF}}(\sigma)$ and then use the DR algorithm in order to achieve the claimed competitive ratio. First, by Lemma 11, the E ϵ'-easy bins can be converted into at most $4/3 \cdot E$ ϵ'-desirable bins (with some additive constant number of bins). Next, by Lemma 12, the H ϵ'-hard bins can be converted into at most $3/2 \cdot H$ ϵ'-desirable bins.

From Lemma 8, DR can be used to approximate P_1 in an online manner by opening at most $(1.5\gamma + 4/3 + \epsilon')E + c$ bins (where c is a constant). Note that the cost of Opt is $H + E = (1+\gamma)E$. Hence, the competitive ratio of the online algorithm is at most $\frac{1.5\gamma+4/3+\epsilon'}{1+\gamma} = \frac{9\gamma+8}{6\gamma+6} + \epsilon$. Further, from Lemma 8, the advice used is $O(2^{3.7/\epsilon'} \cdot \log(1/\epsilon')) = O(2^{3.7(1+\gamma)/\epsilon} \cdot \log(\gamma/\epsilon))$. □

Next, we define the packing P_2 based on $P^{\text{OFF}}(\sigma)$ and the online algorithm that approximates it. Recall that P_{2a} is a packing of the items with size at least $1/3$ and P_{2b} is an ϵ-desirable packing of the items with size no more than $1/3$. To approximate P_{2a}, since it consists of items of size larger than $1/3$, we use Abf (Section 3.1). To approximate P_{2b}, since all the bins are ϵ-desirable, we use the online algorithm DR. This defines an online algorithm that opens at most $((1.3904+3\epsilon')\gamma+1.8349+\epsilon') \cdot E$ bins. The formal details can be found in the proof of the following lemma.

Lemma 14 *There is an online algorithm that uses advice of size $O(2^{11.1/\epsilon} \log(1/\epsilon))$ and achieves a competitive ratio of $\frac{1.3904\gamma+1.8349}{\gamma+1} + \epsilon$.*

Proof Given the packing $P^{\text{OFF}}(\sigma)$, we first create the new packing P_2 consisting of two distinct packings P_{2a} and P_{2b}. The packing P_{2a} consists of the items larger than $1/3$ and P_{2b} consists of the items smaller than or equal to $1/3$ and forms an ϵ'-desirable packing with parameter $\epsilon' = \epsilon \cdot (\gamma+1)/(3\gamma+1)$. To transform the packing $P^{\text{OFF}}(\sigma)$ into P_2, we first consider the ϵ'-hard bins and then the ϵ'-easy bins.

ϵ'-hard bins: For the H ϵ'-hard bins, the total size of items smaller than $1/3$ in each bin is at most ϵ'. The items smaller than $1/3$ from each one of the $\frac{1}{\epsilon'-1}$ ϵ'-hard bins can be packed into a new bin with an empty space of size more than ϵ'. This creates $\frac{\epsilon'}{1-\epsilon'}H < 2\epsilon' H$ new bins that are included in P_{2b}. Note that these new bins include an empty space of size $\epsilon > \epsilon'$ and are ϵ'-desirable of class 0. Further, the original H ϵ'-hard bins now contain only items of size at least $1/3$ and are included in P_{2a}.

ϵ'-easy bins: We split the E ϵ'-easy bins into four sets as follows.

E_0: No items of of size more than $1/3$.
E_{1a}: A single item in the range $(1/3, 1/2]$.
E_{1b}: A single item in the range $(1/2, 1]$.

E_2: Two items in the range $(1/3, 1]$.

For each class of ϵ'-easy bins, the items are repacked such that some of the bins belong to the packing P_{2a} and others belong to the packing P_{2b} as follows.

E_0: By Lemma 11, the bins of E_0 can be repacked into $4/3 \cdot |E_0|$ ϵ'-desirable bins that are included in P_{2b}.

E_{1a}: If the parity of E_{1a} is odd, we add an empty bin to the set E_{1a}. For each pair of bins in E_{1a}, the two items of size in the range $(1/3, 1/2]$ are moved into a new bin that is included in P_{2a}. The original bins are class 0 ϵ'-desirable bins and included in P_{2b}. Overall, this will contribute at most $1/2 \cdot |E_{1a}| + 1$ bins to P_{2a} and $|E_{1a}|$ bins to P_{2b}.

E_{1b}: If the parity of E_{1b} is odd, we add an empty bin to the set E_{1b}. For every pair of bins in E_{1b}, the items smaller than $1/3$ are packed into a new bin. After packing the new bins, by Lemma 11, the items can be repacked into $4/3 \cdot 1/2 \cdot |E_{1b}| = 2/3 \cdot |E_{1b}|$ bins that are included in P_{2b}. The original bins only contain items with size greater than $1/3$ and are included in P_{2a}.

E_2: Up to two empty bins are added to E_2 so that $|E_2| \mod 3 = 0$. For every three bins in E_2, all the items with size less than $1/3$ are packed in a new bin. Similar to E_{1b}, after packing the new bins, by Lemma 11, the items can be repacked into $4/3 \cdot 1/3 \cdot |E_2| = 4/9 \cdot |E_2|$ bins that are included in P_{2b}. The original bins only contain items with size greater than $1/3$ and are included in P_{2a}.

Approximating P_2: Overall, (omitting the additive constants) the number of bins in the new packing is:

$$\underbrace{H + 1/2 \cdot E_{1a} + E_{1b} + E_2}_{P_{2a}\text{: bins with items}>1/3} + \underbrace{2\epsilon' H + E_{1a} + 2/3 \cdot E_{1b} + 4/9 \cdot E_2 + 4/3 \cdot E_0}_{P_{2b}\text{: }\epsilon'\text{-desirable bins with items}\leq 1/3}$$

Note that in the above packing, all the items with size more than $1/3$ are in P_{2a} and the rest are in P_{2b}. Hence, the size of an item determines to which packing it belongs. The algorithm applies ABF to place items of size larger than $1/3$ separately from other items. The result will be an approximation of P_{2a} and requires 1 bit of advice. By Theorem 2, at most $1.3904 \cdot (H + 1/2 \cdot E_{1a} + E_2 + E_0) + E_{1b}$ bins are opened for items larger than $1/3$. Note that E_{1b} corresponds to the set H in Theorem 2. In order to pack the items of P_{2b}, the algorithm uses DR. From Lemma 8, DR will open at most $(1+\epsilon')(2\epsilon' H + E_{1a} + 1/2 \cdot E_{1b} + 1/3 \cdot E_2 + E_0)$ bins and uses advice of size $O(2^{3.7/\epsilon'} \log(1/\epsilon')) = O(2^{11.1/\epsilon} \log(1/\epsilon))$ for $\epsilon' = \epsilon \cdot (\gamma + 1)/(3\gamma + 1) > \epsilon/3$.

The number of bins in the packing of the online algorithm (omitting additive constants) is at most:

$$\begin{aligned}
&1.3904(H + 1/2 \cdot E_{1a} + E_2) + E_{1b} + (1 + \epsilon') \cdot (2\epsilon' H + E_{1a} + 2/3 \cdot E_{1b} + 4/9 \cdot E_2 + 4/3 \cdot E_0) \\
&\leq (1.3904 + 3\epsilon')H + (1.6952 + \epsilon')E_{1a} + (1.67 + \epsilon')E_{1b} + (1.8349 + \epsilon')E_2 + (1.334 + \epsilon')E_0 \\
&\leq (1.3904 + 3\epsilon')H + (1.8349 + \epsilon') \cdot E \\
&= ((1.3904 + 3\epsilon')\gamma + 1.8349 + \epsilon') \cdot E
\end{aligned}$$

We conclude that the competitive ratio of the algorithm is at most

$$\frac{(1.3904+3\epsilon')\gamma+1.8349+\epsilon'}{\gamma+1}=\frac{1.3904\gamma+1.8349}{\gamma+1}+\epsilon.$$

□

We are now ready to prove the main result of this section that shows that advice of constant size (with respect to the length of the input sequence) is sufficient to achieve a competitive ratio arbirarily close to 1.47012.

Theorem 3 *Algorithm* DR+ *uses advice of constant size (dependant on ϵ) and has a competitive ratio of at most* $1.47012+\epsilon$.

Proof We consider two cases depending on the value of γ. Define $\gamma^* := 5015/1096 \approx 4.7633$. If $\gamma \leq \gamma^*$, then, using DR, we approximate the packing P_1 as described in Lemma 13; this gives a ratio of at most $\frac{9\cdot\gamma^*+8}{6\cdot\gamma^*+6}+\epsilon < 1.470112+\epsilon$. If $\gamma > \gamma^*$, then, using DR and ABF, we approximate the packing P_2 as described in Lemma 14; the competitive ratio is at most $\frac{1.3904\cdot\gamma^*+1.8349}{\gamma+1}+\epsilon < 1.47012+\epsilon$. In both cases, the advice is of constant size (Lemmas 13 and 14) that depends on ϵ. □

4 A 7/6 lower bound for sublinear-sized advice

In this section, we prove that any online algorithm with $o(n)$ bits of advice has a competitive ratio of at least 7/6. Our construction is inspired by the one given in [13], which showed a lower bound of 9/8. Our main contribution is a different charging scheme from the one used in [13]. Both lower bounds use a reduction from a variant of the *binary string guessing problem* (2-SGKH) [16,7]. Let $e(X)$ be the number of bits required to encode a number X in a self-delimited fashion. One way to do that is to write the value of $\lceil\log(\lceil\log(X+1)\rceil+1)\rceil$ in unary, the value of $\lceil\log(X+1)\rceil$ in binary, and the value of X in binary. The resulting self-delimited code will have length $e(X) = \lceil\log(X+1)\rceil + 2\lceil\log(\lceil\log(X+1)\rceil+1)\rceil+1$.

In 2-SGKH, the online algorithm must guess an n-length bitstring bit-by-bit. The value of each bit is revealed after the algorithm makes its guess and the algorithm incurs a cost of 1 for each incorrect guess. For a given value of α in the range $[1/2,1)$, let $b_\alpha(n) = (1+(1-\alpha)\log(1-\alpha)+\alpha\log\alpha)n$. Note than $b(n)$ is a linear function of n.

Lemma 15 ([7]) *Any online deterministic algorithm for* 2-SGKH *that is guaranteed to guess correctly more than αn bits, for $1/2 \leq \alpha < 1$, requires at least $b_\alpha(n)$ bits of advice.*

We use the *binary string guessing problem with promise* (2-SGKH$_\beta$) that is parameterized by β. This problem is the same as 2-SGKH except that the input string is guaranteed to have exactly a β fraction of 0s (i.e., βn in total).

Intuitively, this restriction reduces the space of possible request sequences thus making the problem easier in some sense; however, as shown in the following lemma, a linear amount of advice is unavoidable in order to guess more than $\max\{\beta, 1-\beta\}$ bits correctly.[1]

Lemma 16 *Any deterministic algorithm for* 2-SGKH$_\beta$ *that is guaranteed to guess correctly more than* αn *bits, for* $\max\{\beta, 1-\beta\} < \alpha < 1$*, requires at least* $b_\alpha(n)$ *bits of advice, where* $\gamma = \min\{\beta, 1-\beta\}$*.*

Proof The proof is very similar to the proof of Lemma 9 in [13]. Suppose, by way of contradiction, that there is a family of algorithms A_β for 2-SGKH$_\beta$ that correctly guess more than αn bits and uses fewer than $b(n)$ advice bits. We will show how to use A_β so as to obtain an online algorithm B for the 2-SGKH problem. The initial $e(\gamma n)+1$ bits of the advice tape for B contain a bit that indicates whether 0 or 1 is the least frequent bit in the bitstring, as well as $e(\gamma n)$ bits for encoding the number of bits of the least frequent value. With this information, B is able to determine β for this instance of 2-SGKH. The remainder of the advice string contains the bits required by A_β to solve the instance of 2-SGKH. Algorithm B runs A_β on this instance and outputs the guesses of A_β and, therefore, B will be correct whenever A_β is correct. By our initial assumption, B is correct on more than αn bits and uses fewer than $b(n)+e(\gamma n)+1 = (1+(1-\alpha)\log(1-\alpha)+\alpha\log\alpha)n$ bits of advice in total. However, by Lemma 15, this is a contradiction and the lemma follows. □

Let $\mathbb{B}$ denote any algorithm for the bin packing problem; we will show how to obtain an online algorithm $\mathbb{A}$ for 2-SGKH$_{1/2}$ that constructs a request sequence online based on its input and uses the output of $\mathbb{B}$ on this sequence so as to generate its output. Given an instance I of the 2-SGKH$_{1/2}$ problem with a bitstring of length n, we construct a request sequence σ_I for the online bin packing problem with length $2n$ as follows. The sequence consists of a *prefix* of $n/2$ items, a *central part* of n items and a *suffix* of $n/2$ items. All $n/2$ items of the prefix have a size of $1/2+\epsilon$, where ϵ is an arbitrary small positive value. The n central items have distinct sizes in the range $(1/2-2\epsilon, 1/2-\epsilon)$. (The exact manner in which their size is determined is explained subsequently.) Among these n items, we refer to the smallest $n/2$ items as *small* items and to the remaining items as *large* items. We emphasize that this terminology is independent of Section 2. The $n/2$ items of the suffix are the exact complements of the small items, i.e., for any small item of size x, there is an item of size $1-x$ in the suffix. We observe that there is a packing of σ that uses n bins. The $n/2$ small items are packed with their complements in the suffix; moreover, the $n/2$ large items are packed each with an item of the prefix. Since there are n items of size strictly more than $1/2$, this packing is optimal.

[1] Technically, the statement of Lemma 16 is very similar to Lemma 9 in [13]. We note, however, that the latter is correct only when the number of 0s is $n/2$. To avoid any ambiguity, the statement of Lemma 16 is parameterized by β, as opposed to Lemma 9 in [13].

The size of each of the n central items of σ is determined as in [13], using an iterative process. More precisely, we do so by maintaining an interval which is initially set to $(1/2 - 2\epsilon, 1/2 - \epsilon)$. The endpoints of the interval (except the initial values) indicate the largest small item and the smallest large item among the revealed items. The size of the next item is in the middle of the interval, e.g., the first item has size $1/2 - 1.5\epsilon$. Let b_i be the i-th such item and let the current interval be (x, y). Then, the size of b_i is $(x+y)/2$. If $r_i \in \sigma_{\mathcal{B}}$ is 0, then b_i is defined to be a small item and the interval is updated to $((x + y)/2, y)$. Otherwise, r_i is 1, b_i is defined to be a large item, and the interval is updated to $(x, (x + y)/2)$. Note that the size of the item b_{i+1} reveals to the algorithm $\mathbb{B}$ whether b_i was small or large. This relates back to 2-SGKH$_{1/2}$, where the bit values are revealed after the guesses.

Note that $\mathbb{B}$ must open a bin for the $n/2$ items of the prefix of $\sigma_{\mathcal{B}}$ as they all have a size greater than $1/2$. The manner in which $\mathbb{B}$ packs each of n central items will determine the n guesses of $\mathbb{A}$. Let b_i be the i-th such item. Algorithm $\mathbb{B}$ has 3 options for packing b_i: (1) to open a new bin for b_i; (2) to pack b_i in a bin with an item from the prefix; or (3) to pack b_i in a bin with some item b_j, $j < i$. If $\mathbb{B}$ chooses option (1), the item is labelled as small and $\mathbb{A}$ guesses 0. If $\mathbb{B}$ chooses either option (2) or (3), the item is labelled as large and $\mathbb{A}$ guesses 1.

The following lemma relates the number of incorrect guesses (or number of mislabeled items) to the number of extra bins opened (in comparison to Opt). We will use the same accounting technique as in [13], but a new mapping of incorrect guesses to mislabellings, which leads to an improved bound. More precisely, we show that each extra bin corresponds to 3 mislabellings (as opposed to [13], in which the corresponding number equals 4). Let f_n denote the family of all possible request sequences $\sigma_{\mathcal{B}}$, as described above, for all possible bitstrings of length n with exactly $n/2$ 0s.

Lemma 17 *Suppose that there is an algorithm $\mathbb{B}$ that uses $b(n)$ bits of advice and opens at most $\textsc{Opt}(\sigma) + c$ bins for all $\sigma \in f_n$. Then, there exists an algorithm for the 2-SGKH$_{1/2}$ problem that uses $b(n)$ bits of advice and makes at most $3c$ errors.*

Proof Let $\mathbb{A}$ be the online algorithm described above that uses $\mathbb{B}$ to solve instances of the 2-SGKH$_{1/2}$ problem. Note that in the reduction a mislabelling of an item is equivalent to a wrong guess by $\mathbb{A}$. Thus, we can relate the number of errors of $\mathbb{A}$ to the additional bins opened by $\mathbb{B}$ as compared to $P^{OPT}(\sigma)$ by means of the mislabeled items.

We observe that, without loss of generality, $\mathbb{B}$ places the items of the suffix in a bin with its complement, if possible, otherwise it opens a new bin. In addition, we can assume that every small item, for which $\mathbb{B}$ opened a bin, will be packed with its complement. As a consequence, the additional bins opened by $\mathbb{B}$ are exactly those that were opened for a large item.

Given the three options of the algorithm, we distinguish between 3 types of mislabellings: (1) a large item opens a new bin; (2) a small item is packed

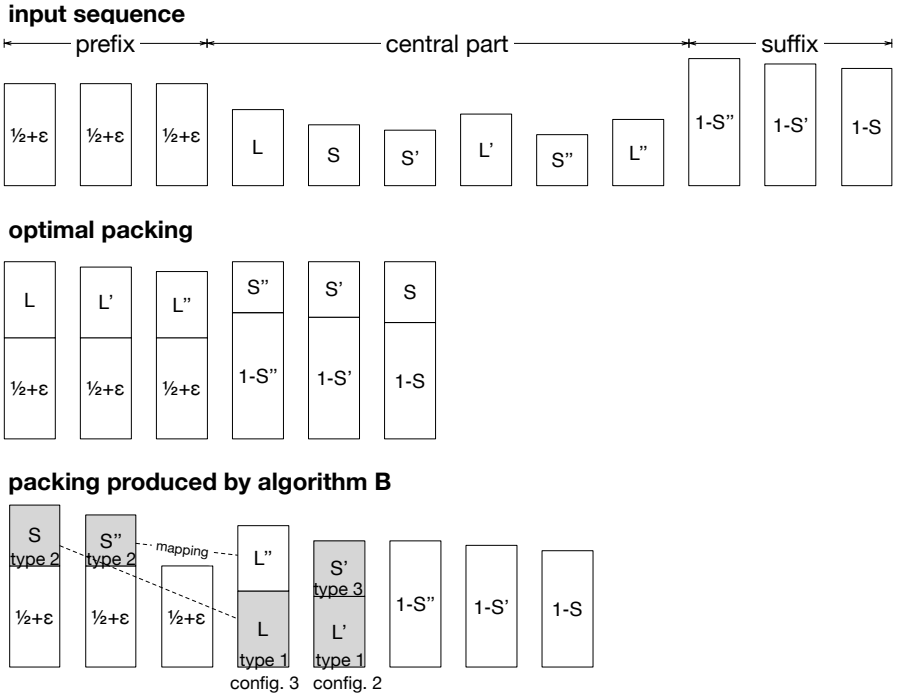


Fig. 3 Illustration of the lower-bound construction. The top figure shows sequence σ, with the optimal packing depicted in the middle figure. The bottom figure depicts a packing produced by $\mathbb{B}$ and illustrates the mislabellings and the mappings defined in the proof of Lemma 17. Mislabelled items are colored gray.

in a bin with a prefix item; and (3) a large item is packed with a small item (see Figure 3 for an illustration).

Next, we show how to map every mislabelling to an additional bin (in comparison to $P^{OPT}(\sigma)$) in such a way that at most 3 mislabbelings are mapped to each additional bin.

First, we match all mislabellings of type 2 with large items that are not packed with prefix items, see figure 3. This is possible because for every large item not packed with a prefix item, there is a prefix item not packed with a large item, and some of those may be packed with a small item.

Second, we assign mislabellings to additional bins, as follows. A type 1 mislabelling is mapped to the bin opened for the large item. A type 2 mislabelling is assigned to the additional bin hosting the large item to which it is mapped. A type 3 mislabelling is mapped to the bin containing the corresponding small item.

Third, we consider three different configurations of the additional bins opened by $\mathbb{B}$, and count the number of mislabellings assigned to them. All configurations contain a large item, for which the bin was opened, which is (1) alone; or (2) packed with a small item; or (3) packed with a large item.

All configurations correspond to a distinct type 1 mislabelling, plus possibly a type 2 mislabelling. In configuration (2), we have to count an additional type 3 mislabelling. In configuration (3), the second large item (i.e., the item that was placed last in the bin) is correctly labelled, but there may be a second type 2 mislabelling assigned to it.

In summary, there is a mapping from mislabellings to additional bins, such that every bin is the image of at most 3 mislabellings. This completes the proof of the lemma. □

We can now show that $\Omega(n)$ advice bits are necessary to obtain a competitive ratio better than 7/6.

Theorem 4 *Any deterministic online algorithm with advice for the bin packing problem requires at least* $b_\alpha(n)-e(n/2)-1$ *bits of advice to be* ρ*-competitive,* $1 < \rho < 7/6$*, where* $\alpha = 4 - 3\rho$*.*

Proof Let $\mathbb{B}$ be an algorithm for the bin packing problem with a competitive ratio of ρ that uses $b(m)$ bits of advice, where m is the length of the request sequence. For a request sequence σ and $\textsc{Opt}(\sigma) = n$, $\mathbb{B}$ uses at most $\rho n = n + (\rho - 1)n$ bins. By Lemma 17, this implies that there exists an algorithm $\mathbb{A}$ for the 2-SGKH$_{1/2}$ problem that uses $b(2n)$ bits of advice and makes at most $3(\rho - 1)$ errors on an input string of length n. That is, $\mathbb{A}$ is correct on $n - 3(\rho - 1)n = (4 - 3\rho)n$ bits. For $\alpha = (4 - 3\rho)$, the bounds on ρ imply that $1/2 < \alpha < 1$. The claim follows by applying Lemma 16 with $\alpha = (4 - 3\rho)$ and $\beta = 1/2$. □

5 Conclusion

In this work we studied the effect of constant-size advice on the performance of online bin packing algorithms. On the positive side, we introduced and analyzed two different online algorithms: a relatively simple algorithm (RedBlue) with competitive ratio $1.5+\epsilon$, and a more complex algorithm (DR+) with competitive ratio $1.47012 + \epsilon$. We note that algorithm RedBlue converges to 1.5 quicker than the more complicated algorithm DR+; in particular, RedBlue outperforms all online algorithms with 16 bits of advice. On the negative side, we showed that advice of linear size is required to achieve a competitive ratio better than 7/6.

The obvious direction for future works is to improve the upper and lower bounds on the competitive ratio. For the upper bound, we note that that breaking the barrier of 1.5 on the competitiveness of online algorithms (and thus also the bound established in this paper) may indeed require an involved approach. More precisely, even for the special case in which all items have size greater than 1/3, almost all classic bin packing algorithms such as BestFit, NextFit, FirstFit, Harmonic have competitive ratio equal to 1.5. For the lower bound, it would be interesting to consider a further strengthening of the reduction from the string guessing problem. More precisely, it would

be interesting to apply the weighted counting scheme of Mikkelsen [27] to a different request sequence than the one used in this paper as well as in [13]. For instance, one could consider request sequences similar to the one of Section 4 which, however, do not have a prefix of items of size larger than 1/2. This would be beneficial, since the cost of the optimal solution would not be inflated by the presence of such items.

References

1. Adamaszek, A., Renault, M.P., Rosén, A., van Stee, R.: Reordering buffer management with advice. Journal of Scheduling (2016). DOI 10.1007/s10951-016-0487-8
2. Angelopoulos, S., Dürr, C., Kamali, S., Renault, M.P., Rosén, A.: Online bin packing with advice of small size. In: F. Dehne, J. Sack, U. Stege (eds.) Algorithms and Data Structures - 14th International Symposium, WADS 2015, Victoria, BC, Canada, August 5-7, 2015. Proceedings, *Lecture Notes in Computer Science*, vol. 9214, pp. 40–53. Springer (2015). DOI 10.1007/978-3-319-21840-3_4
3. Asgeirsson, E., Ayesta, U., Coffman, E., Etra, J., Momcilovic, P., Phillips, D., Vokhshoori, V., Wang, Z., Wolfe, J.: Closed on-line bin packing. Acta Cybernet. **15**(3), 361–367 (2002)
4. Balogh, J., Békési, J., Galambos, G.: New lower bounds for certain classes of bin packing algorithms. Theoret. Comput. Sci. **440–441**, 1–13 (2012)
5. Bianchi, M.P., Böckenhauer, H., Brülisauer, T., Komm, D., Palano, B.: Online minimum spanning tree with advice - (extended abstract). In: R.M. Freivalds, G. Engels, B. Catania (eds.) SOFSEM 2016: Theory and Practice of Computer Science - 42nd International Conference on Current Trends in Theory and Practice of Computer Science, Harrachov, Czech Republic, January 23-28, 2016, Proceedings, *Lecture Notes in Computer Science*, vol. 9587, pp. 195–207. Springer (2016). DOI 10.1007/978-3-662-49192-8_16
6. Böckenhauer, H., Komm, D., Královic, R., Rossmanith, P.: The online knapsack problem: Advice and randomization. Theor. Comput. Sci. **527**, 61–72 (2014). DOI 10.1016/j.tcs.2014.01.027
7. Böckenhauer, H.J., Hromkovic, J., Komm, D., Krug, S., Smula, J., Sprock, A.: The string guessing problem as a method to prove lower bounds on the advice complexity. Theoret. Comput. Sci. **554**, 95–108 (2014)
8. Böckenhauer, H.J., Komm, D., Královič, R., Královič, R.: On the advice complexity of the k-server problem. In: Proc. 38th International Colloquium on Automata, Languages, and Programming (ICALP), *Lecture Notes in Comput. Sci., Springer*, vol. 6755, pp. 207–218 (2011)
9. Böckenhauer, H.J., Komm, D., Královič, R., Královič, R., Mömke, T.: On the advice complexity of online problems. In: Proc. 20th International Symp. on Algorithms and Computation (ISAAC), *Lecture Notes in Comput. Sci., Springer*, vol. 5878, pp. 331–340 (2009)
10. Borodin, A., El-Yaniv, R.: Online Computation and Competitive Analysis. Cambridge University Press (1998)
11. Boyar, J., Favrholdt, L., Kudahl, C., Larsen, K.S., Mikkelsen, J.W.: Online algorithms with advice: A survey. SIGACT News **47**(3), 93–129 (2016)
12. Boyar, J., Kamali, S., Larsen, K.S., López-Ortiz, A.: On the list update problem with advice. Information and Computation (2016). DOI 10.1016/j.ic.2016.06.007
13. Boyar, J., Kamali, S., Larsen, K.S., López-Ortiz, A.: Online bin packing with advice. Algorithmica **74**(1), 507–527 (2016). DOI 10.1007/s00453-014-9955-8
14. Dobrev, S., Kralovic, R., Pardubská, D.: How much information about the future is needed? In: Proc. 34th International Conf. on Current Trends in Theory and Practice of Computer Science (SOFSEM), *Lecture Notes in Comput. Sci., Springer*, vol. 4910, pp. 247–258 (2008)
15. Dobrev, S., Královič, R., Pardubská, D.: Measuring the problem-relevant information in input. RAIRO Inform. Theor. Appl. **43**(3), 585–613 (2009)

16. Emek, Y., Fraigniaud, P., Korman, A., Rosén, A.: Online computation with advice. Theoret. Comput. Sci. **412**(24), 2642 – 2656 (2011)
17. Epstein, L., Levin, A.: On bin packing with conflicts. SIAM J. Optim. **19**(3), 1270–1298 (2008)
18. Galambos, G., Woeginger, G.J.: Repacking helps in bounded space online bin packing. Computing **49**, 329–338 (1993)
19. Gambosi, G., Postiglione, A., Talamo, M.: Algorithms for the relaxed online bin-packing model. SIAM J. Comput. **30**(5), 1532–1551 (2000)
20. Garey, M.R., Graham, R.L., Ullman, J.D.: Worst-case analysis of memory allocation algorithms. In: P.C. Fischer, H.P. Zeiger, J.D. Ullman, A.L. Rosenberg (eds.) STOC, pp. 143–150. ACM (1972)
21. Garey, M.R., Johnson, D.S.: Approximation algorithms for bin packing problems - a survey. In: G. Ausiello, M. Lucertini (eds.) Analysis and Design of Algorithms in Combinatorial Optimization, pp. 147–172. Springer, New York (1981)
22. Grove, E.F.: Online binpacking with lookahead. In: Proc. 6th Symp. on Discrete Algorithms (SODA), pp. 430–436 (1995)
23. Gupta, S., Kamali, S., López-Ortiz, A.: On advice complexity of the k-server problem under sparse metrics. Theory Comput. Syst. **59**(3), 476–499 (2016)
24. Heydrich, S., van Stee, R.: 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, pp. 41:1–41:14 (2016)
25. Johnson, D.S.: Near-optimal bin packing algorithms. Ph.D. thesis, MIT, Cambridge, MA (1973)
26. Johnson, D.S., Demers, A., Ullman, J.D., Garey, M.R., Graham, R.L.: Worst-case performance bounds for simple one-dimensional packing algorithms. SIAM J. Comput. **3**, 256–278 (1974)
27. J.W., M.: Randomization can be as helpful as a glimpse of the future in online computation. In: 43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy, pp. 39:1–39:14 (2016)
28. Komm, D., Královič, R.: Advice complexity and barely random algorithms. RAIRO Inform. Theor. Appl. **45**(2), 249–267 (2011)
29. Renault, M.P., Rosén, A.: On online algorithms with advice for the k-server problem. Theory Comput. Syst. **56**(1), 3–21 (2015). DOI 10.1007/s00224-012-9434-z
30. Renault, M.P., Rosén, A., van Stee, R.: Online algorithms with advice for bin packing and scheduling problems. Theor. Comput. Sci. **600**, 155–170 (2015). DOI 10.1016/j.tcs.2015.07.050
31. Sloane, N.J.A.: The on-line encyclopedia of integer sequences. Sequence A000041.