Online Bin Packing with Advice of Small Size

Spyros Angelopoulos[1,2,*], Christoph Dürr[1,2,*], Shahin Kamali[3,*,†], Marc Renault[1,*], and Adi Rosén[4,*]

[1] Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, F-75005, Paris, France
[2] CNRS, UMR 7606, LIP6, F-75005, Paris, France
[3] University of Waterloo, Waterloo, Canada.
[4] CNRS and Université Paris Diderot, Paris, France.

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 which is below 1.5-competitive, namely has competitive ratio arbitrarily close to 1.47012. This is the currently best performance of any online bin packing algorithm with sublinear advice. On the negative side, we extend a construction due to Boyar *et al.* [10] so as to show that no online algorithm with sub-linear advice can be 7/6-competitive, which improves upon the known lower bound of 9/8.

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, 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, we are interested in the *asymptotic* competitive ratio which considers sequences for which the costs of $\mathbb{A}$ and OPT are arbitrarily large.

*Research supported in part by project ANR-11-BS02-0015 "New Techniques in Online Computation–NeTOC".

†Research supported by the France Canada Research Fund.

For this reason, throughout this paper we refer to the asymptotic competitive ratio as simply the competitive ratio.

The bin packing problem has provided some of the first-known explicit online algorithms. 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 Next Fit is 2-competitive, whereas FIRSTFIT and BESTFIT are both 1.7-competitive [18]. The best known online algorithm is HARMONIC++ which has a competitive ratio of 1.588 [22]. No online algorithm can have a competitive ratio better than 1.54037 [3], 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 [8]. Different models have been proposed in order to address this issue, and one such approach is by allowing the online algorithm certain additional power. For example, the algorithm may be allowed to repack some items [14,15]. Alternatively, it may have access to lookahead [16], and, finally, may know the length of the input sequence [2] or the value of OPT [13]. The advice model is a generalization of the latter 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 emphasize that the advice setting may in fact have tangible applications. For instance, the advice model captures, among others, any relevant statistical information about the input that may be available through either preprocessing or historical data. We define the bin packing problem under the advice setting as follows:

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.*

Throughout the paper, for a given algorithm $\mathbb{A}$, we denote by $\mathbb{A}(\sigma)$ the number of bins used by $\mathbb{A}$ on sequence σ. Due to space limitations, we omit or sketch certain proofs (complete proofs can be found in the long version of the paper).

1.1 Previous work and our contribution

The online advice model was first introduced by Böckenhauer *et al.* [7,6] and by Emek *et al.* [12]. Both papers were inspired by the work of Dobrev *et al.* [11]. In the model of Emek *et al.* 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.*, 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 [12], job shop scheduling [7,19], the k-server problem [12,6,20], knapsack [5], buffer reordering management [1], and list update [9]).

In this paper, we study online bin packing under the advice-on-tape model. In this setting, Boyar *et al.* [10] 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.* [21] 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. Boyar *et al.* showed that advice of size $\Theta(\log n)$ is sufficient to achieve an algorithm with competitive ratio of 1.5, which is strictly better than the lower bound 1.54037 for online algorithms. They also proved that no algorithm is better than 9/8-competitive with advice of sub-linear size. A related problem, namely the minimum makespan problem on identical machines was studied in [21].

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 for advice of sublinear size and demonstrates the significant impact of small-size advice on algorithmic performance. We should mention that the simple algorithm of Section 2 reaches $1.5 + \epsilon$ with fewer advice bits than the complicated algorithm of Section 3. Last, we give a lower-bound construction that builds on ideas of [10] 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, we introduce two algorithms that, when combined, result in the desired upper bound. One of these algorithms uses a rounding technique to create close-to-optimal packings when there is an empty space of size ϵ or more in all bins of an optimal solution (ϵ is an arbi-

trary small positive value). The other algorithm achieves a competitive ratio of 1.3904 when all items are larger than 1/3. Both algorithms use advice of constant size, i.e., independent of the length of sequence. Last, concerning the lower bound (Section 4), we base our construction on that of [10], using a better amortization scheme that leads to an improvement of the bound.

2 Constant-size advice outperforms all online algorithms

In this section, we present an algorithm that achieves a competitive ratio of $1.5 + \epsilon$ and uses a constant number of bits of advice. Throughout the section, we distinguish items based on their sizes. An item is *huge* if it is larger than 2/3, *critical* if it is in the range $(1/2, 2/3]$, *small* if it is in the range $(1/3, 1/2]$, and *tiny* if it is in the range $(0, 1/3]$.

Consider the algorithm RESERVECRITICAL [10] that works as follows. The algorithm treats huge items separately and places each of them in a single bin. Similarly, it places two small items in the same bin with no other items in said bin. The algorithm knows the number of critical items and reserves space of size $2/3$ for each of them (i.e., it opens a bin for each item and assumes the filled space of the bin is 2/3). Critical items are placed in the reserved spaces. For tiny items, the algorithm uses FIRSTFIT to place them in critical bins with respect to their reserved spaces (and opens new bins as needed). To encode the number of critical bins in binary, $\Theta(\log n)$ advice bits are needed. As shown in [10], RESERVECRITICAL has a competitive ratio of 1.5. (Since our approach is related, in the long version of the paper, we provide a simpler proof of the result in [10].)

In what follows, we analyze another algorithm, called the REDBLUE algorithm, that receives an integer i, $0 \leq i < 2^k$ encoded in binary with 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. Let X and Y denote the number of bins in the packing of RESERVECRITICAL that include a critical item, and the number of bins opened for the tiny items, respectively. The advice encodes an approximate value of $\frac{X}{X+Y}$, using k bits, by encoding the value of i such that

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

Regardless of the value of β, REDBLUE always places each huge item in a single bin, and places small items in dedicated bins, with two such items per dedicated bin. In the following, we consider three (exhaustive) cases for β: $\beta > 1 - 1/2^{k/2}$, $\beta < 1/2^{k/2}$, and $1 - 1/2^{k/2} \leq \beta \leq 1/2^{k/2}$. For each case, we complete the definition of REDBLUE by describing how the critical and tiny items are packed.

Consider the first case: $\beta > 1 - 1/2^{k/2}$. For placing critical and tiny items, REDBLUE maintains a set of *blue* bins such that each bin has a reserved space of 2/3 for critical items. To pack a critical item, REDBLUE packs it using FIRSTFIT among the set of blue bins, considering only the reserved space. To pack a tiny item, REDBLUE packs it using FIRSTFIT among the set of blue bins, considering, however, only the non-reserved space (of size 1/3) of such bins. Any bin that FIRSTFIT opens for critical and tiny items will be blue, i.e., it has a reserved space of 2/3 for critical and a space of 1/3 for tiny items.

Lemma 1. *When $\beta > 1 - 1/2^{k/2}$, the competitive ratio of the* REDBLUE *algorithm is at most* $1.5 + \frac{7.5}{2^{k/2}}$.

Proof. From (1) and the statement of the lemma, we have:

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

Let B denote the set of blue bins. The first X bins in B are precisely the first X bins in the packing of RESERVECRITICAL, i.e., they include X critical items plus the same tiny items. Let Y' denote the number $|B| - X$. Then Y' bins in B only include tiny items (i.e., the reserved space is not occupied by a critical item); the level of all these bins, except possibly one, is at least 1/6 (otherwise, FIRSTFIT could combine two in the same bin). Since the tiny items placed in these bins are the same as those placed in the last Y bins of the RESERVECRITICAL algorithm, we have $Y' \leq 6Y + 1$; this is because the level of bins in the REDBLUE packing is at least 1/6. Let H and S denote the number of huge and small items. From (2) and the fact that $\text{RESERVECRITICAL}(\sigma) = H + \lceil S/2 \rceil + X + Y \leq 1.5\,\text{OPT}(\sigma)$, we obtain $\text{REDBLUE}(\sigma) \leq H + \lceil S/2 \rceil + X + 6Y + 1 < (1 + 5/2^{k/2}) \cdot 1.5\,\text{OPT}(\sigma) + 1$. □

Next, we consider the second case: $\beta < 1/2^{k/2}$. In this case, REDBLUE maintains a set of blue bins for critical items and a set of *red* bins for tiny items. The algorithm applies FIRSTFIT to pack critical items into the set of blue bins and tiny items into the set of red items. In this case, all the bins except the blue bins have a level of at least 2/3; moreover, there are only a few blue bins. We can show that, on average, the level of all the bins (except 1 bin) is very close to 2/3.

Lemma 2. *When $\beta < \frac{1}{2^{k/2}}$, the competitive ratio of* REDBLUE *is at most* $3/2 + \frac{3}{2^k - 2}$.

Next, we focus on the case $1 - 1/2^{k/2} \leq \beta \leq 1/2^{k/2}$. In this case, the algorithm maintains a set of blue bins such that each bin has a reserved space of 2/3 for critical items. The remaining unreserved space of 1/3 will be used for packing tiny items. The algorithm also maintains a set of red bins for packing tiny items.

We now explain precisely how REDBLUE packs critical and tiny items. For a critical item x, REDBLUE uses FIRSTFIT among the blue bins, and places x in the reserved space of such a bin. If x opens a new bin, the bin is declared blue. For a tiny item y, the algorithm applies FIRSTFIT to place y in either the unreserved space of a blue bin, or in a red bin. If the algorithm cannot place y in one of the existing bins, it opens a new bin for y. It declares the new bin as either a red or a blue bin as follows. Let B and R denote the number of blue and red bins, immediately before y is packed, respectively. The algorithm will then declare the new bin as a blue bin if $\frac{B+1}{B+R+1} \leq \beta$; otherwise, it will declare the new bin as red. Note that, in this way, REDBLUE guarantees that $\frac{B_n}{B_n + R_n} \leq \beta$, where B_n and R_n denote the number of blue and red bins after processing the entire sequence, respectively. 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, since β is a lower bound for the ratio $\frac{X}{X+Y}$, this strategy ensures that all bins declared as blue eventually receive a critical item.

Lemma 3. *When* $1/2^{k/2} \leq \beta \leq 1 - 1/2^{k/2}$*, the competitive ratio of* RedBlue *is less than* $1.5 + \frac{3}{2^{k/2}-2}$.

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

$$\frac{1-\beta}{\beta} X < \left(1 + \frac{1}{\beta(1-\beta)2^k - \beta}\right) Y < \left(1 + \frac{1}{2^{k/2-1} - 1}\right) Y \tag{3}$$

Let y_i be a tiny item for which RedBlue opens the very last red bin in its packing. Let R_i and B_i denote the number of red and blue bins *after* placing y_i, respectively. From the statement of the algorithm, we have $\frac{B_i+1}{B_i+R_i} > \beta$, which implies that $R_i < \frac{(1-\beta)B_i+1}{\beta}$. Let R_n and B_n be the number of red and blue bins in the final packing of the algorithm. We have $R_n = R_i$ and $B_i \leq B_n = X$. For the given range of β, in the final packing, all blue bins receive a critical item. Hence, $R_n \leq \frac{1-\beta}{\beta} X + 1/\beta$. From the above inequality, we obtain $B_n + R_n \leq X + Y + \left(\frac{2}{2^{k/2}-2}\right) Y + 2^{k/2}$, and the cost of the algorithm can then be bounded as follows: $\textsc{RedBlue}(\sigma) = H + \lceil S/2 \rceil + B_n + R_n \leq H + \lceil S/2 \rceil + X + Y + 2Y/(2^{k/2)} - 2) + 2^{k/2} \leq 1.5\,\textsc{Opt}(\sigma) + \frac{3}{2^{k/2}-2}\,\textsc{Opt}(\sigma) + 2^{k/2}$, where we used (3) and the fact that $\textsc{ReserveCritical}(\sigma) = H + \lceil S/2 \rceil + X + Y \leq 1.5\,\textsc{Opt}(\sigma)$. Note also that $2^{k/2}$ is a constant independent of n. □

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

Proof. From Lemmas 1, 2,3, the competitive ratio of the algorithm is no more than $1.5 + \max\left\{\frac{15}{2^{k/2+1}}, \frac{3}{2^k-2}, \frac{3}{2^{k/2}-2}\right\}$ 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.

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

We will present and analyze an online algorithm with constant number of advice bits that has a competitive ratio that is arbitrarily close to 1.47012. To this end, we will first introduce an algorithm for sequences in which all items are relatively large, namely larger than 1/3. We will then use this algorithm as a subroutine in the final algorithm that handles arbitrary sequences.

3.1 Sequences with items larger than 1/3

Assume all items are larger than $1/3$. We will show that with only 1 bit of advice, we can achieve solutions which are 1.3904-competitive. For the remainder of this subsection, an item is said to be *small* if it is no larger than 1/2, *large* if it has size larger than 1/2 and is placed with a small item in the optimal packing, and *huge* if it is larger than 1/2 and is alone in its bin in the optimal packing. We use S, L and H to denote

the number of small, large and huge items, respectively. We can assume that the size of any huge item is no smaller than large items (otherwise they can be switched and thus obtain another optimal packing). The cost of OPT for the input sequence σ is then $\text{OPT}(\sigma) = H + L/2 + S/2$. We use $\text{OPT}_2(\sigma)$ to denote the number of bins in the optimal packing that include two items, i.e., $\text{OPT}_2(\sigma) = S/2 + L/2$.

The following is the main theorem of this subsection, and will also be used later in the proof of Lemma 10, in the context of general sequences.

Theorem 2. *For a sequence σ in which all items are strictly larger than 1/3, there is an online algorithm with 1 bit of advice that opens at most $H + 1.3904 \cdot \text{OPT}_2(\sigma)$ bins.*

The following result is direct from Theorem 2, observing that $\text{OPT}(\sigma) = H + \text{OPT}_2(\sigma)$.

Corollary 1. *There is an algorithm for online bin packing with items larger than 1/3 that uses 1 bit of advice and that has competitive ratio 1.3904.*

The single advice bit serves the purpose of determining the best algorithm among two purely online algorithms: ALMOSTBESTFIT (ABF) and CROSSBESTFIT (CBF). ABF is similar to BESTFIT except that it opens a new bin for each item larger than 1/2. CBF also applies BESTFIT, but it opens a new bin for each item smaller than or equal to 1/2.

In order to prove Theorem 2, we consider three different cases and show that in each case, at least one of ABF and CBF is better than 1.3904-competitive. To define these cases, we consider two parameters α and β such that $0 \leq \alpha \leq 1$ and $1 \leq \beta < 2$. We will determine the values of these parameters later in the proof. Note that in an optimal packing, L large items are matched with small items. We call such two items *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 the optimal packing. Let $X \leq L$ denote the number of large items which have their partners among the forthcoming items (at the time they are placed). We consider the following three (exhaustive) cases: I) $L \leq \frac{\beta-1}{2-\beta}S$, II) $L > \frac{\beta-1}{2-\beta}S$ and $X \geq \alpha L$, and III) $L > \frac{\beta-1}{2-\beta}S$ and $X < \alpha L$.

In the final packing of ABF, all small items (except potentially one of them) are placed with another item. With this observation, we can prove the following for Case I:

Lemma 4. *If $L \leq \frac{\beta-1}{2-\beta}S$,* ABF *opens at most $H + \beta \cdot \text{OPT}_2(\sigma)$ bins.*

Next, we consider Case II.

Lemma 5. *If $L > \frac{\beta-1}{2-\beta}S$ and $X \geq \alpha L$ then* ABF *opens at most $H + (3/2 - \alpha/2) \cdot \text{OPT}_2(\sigma)$ bins.*

Proof. We claim that in the packing of ABF at least X small items are packed with large items. If this is true, then the number of bins opened by ABF is at most $H + L + (S - X)/2 \leq H + L + (S - \alpha L)/2 = H + (2 - \alpha)L/2 + S/2$. 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 $(2 - \alpha)L/2 + S/2 \leq \frac{3-\alpha}{2}\text{OPT}_2(\sigma)$. We thus conclude that $\text{ABF}(\sigma) \leq H + (3/2 - \alpha/2)\text{OPT}_2(\sigma)$.

It remains to prove the claim. We maintain a mapping of size X as follows formed by X pairs of items. The mapping is initially formed by the X large items and their partners which appear later. We use $m(y)$ to denote the mapped item of an item y. The mapping is said to be *valid* if it has the following properties: i) for any pair $(x, m(x))$ in the mapping, x is larger than 1/2 and $x + m(x) \leq 1$; and ii) for any pair $(x, m(x))$ in the mapping, x appears earlier than $m(x)$ in the sequence. Note that the initial mapping is valid. We will show how to maintain a valid mapping of size X, upon the arrival and packing of each item, in such a way that all pairs of mapped items are placed in the same bin by the ABF algorithm.

Suppose that a new item y arrives. If y is larger than 1/2, a new bin is opened for it and the mapping does not change. Next, suppose that y is small; moreover, suppose that the pair (z, y) is in the current mapping, for some item z. If y is placed with z in the same bin, then the mapping does not change. Assume y is placed with another item z' which is larger than z (by BESTFIT it cannot be placed with a smaller item). If z' is in the mapping, we replace (z, y) with (z', y) (with a slight abuse of notation, we will say that an element r is in the mapping if there is an element q such that the pair (r, q) is in the mapping). Otherwise, since $z \leq z'$ we have $z + m(z') \leq 1$. In this case, we replace (z, y) and $(z', m(z'))$ with (z', y) and $(z, m(z'))$, respectively. The result is still a valid mapping. Finally, suppose that y is smaller than 1/2 and it is not in the mapping. The mapping is not changed after packing y unless y is packed with an item z which is in the mapping. Note that z cannot be small; otherwise, it would have been placed with the large item that it is mapped to upon its arrival. Hence, z is a large item. In this case, we replace the pair $(z, m(z))$ with (z, y); this maintains a valid mapping. □

Finally, it remains to consider Case III. The proof of the following lemma uses techniques similar to the proof of Lemma 5.

Lemma 6. *Suppose $L > \frac{\beta-1}{2-\beta} S$ and $X < \alpha L$ then the number of bins opened by* CBF *is at most $H + (4 - 2(\alpha + \beta) + 2\alpha\beta) \cdot \text{OPT}_2(\sigma)$.*

Proof of Theorem 2. From Lemmas 4, 5, 6, the competitive ratio of the best algorithm among ABF and CBF is at most $\max\{\beta, 3/2 - \alpha/2, 4 - 2(\alpha + \beta) + 2\alpha\beta\}$, where $0 \leq \alpha \leq 1$ and $1 \leq \beta < 2$. The optimal choice is $\beta = (7 + \sqrt{17})/8$ and $\alpha = (5 - \sqrt{17})/4$ which gives a competitive ratio at most $\beta < 1.3904$. □

3.2 Arbitrary sequences

We use the result of the previous section to 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, $0 < \epsilon < 1/12$. To this end, we first define *ϵ-desirable* solutions.

Definition 2. *An* ϵ-desirable packing *of a sequence σ is a packing formed by a set of* ϵ-desirable bins*. A bin is ϵ-desirable and belongs to class 0 if there is an empty space of size at least ϵ in the bin. A bin is ϵ-desirable and belongs to class i ($i \in \{1, 2, 3\}$) if its empty space is less than ϵ and if it includes i items in the range $(1/i - \epsilon, 1/i]$.*

We begin with an outline of our approach. First, we will show that, for any packing that consists of X ϵ-desirable bins, there is an online algorithm DESIRABLEROUDING

(DR) which opens $(1+\epsilon)X$ bins and requires advice of size $f(\epsilon)$, where f is a function of ϵ (Lemma 7). Given an ϵ and an optimal offline packing of a sequence σ, we will define two new packings P_1 and P_2 in such a way that at least one of them provides a good approximation of the optimal packing, and the 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 the DR algorithm. To approximate P_2, we use the algorithm from Section 3.1 so as to approximate P_{2a} and DR so as to approximate P_{2b}. One additional bit of advice can determine the best among the two online approximations of P_1 and P_2.

We now proceed with the technical details of the algorithm.

Lemma 7. *Consider an ϵ-desirable packing* OFF *of a sequence σ. There is an online algorithm* DR *with advice of size $O(2^{3.7/\epsilon} \cdot \log(1/\epsilon))$ that outputs a packing with at most $(1+\epsilon)$ $\textsc{Off}(\sigma)$ bins, where $\textsc{Off}(\sigma)$ is the number of bins in the desirable packing.*

Proof Sketch. We give an outline of the proof. The full details are in the long version of the paper. Given an ϵ-desirable packing, the item sizes are rounded up so that there are m different item sizes or *item types*, where m is inversely proportional to ϵ^2. By applying this rounding scheme, there will be a constant (inversely proportional to ϵ^2) number of possible *bin types*, where the type of a bin is based on the number and types of the rounded items packed within. The advice indicates the approximate value for the fraction of bins from each bin type in the desirable packing. Each of these values are encoded in k bits, where k is function of ϵ. Provided with this advice, DR maintains similar ratios for the bins of each type that it opens. To accomplish this, instead of opening single bins, it opens a family of bins in which the bin types are pre-determined so as to maintain the same fraction of bin types as indicated by the advice. Each arriving item is packed into the appropriate reserved space based on the bin and item types. □

Lemma 7 suggests that we need ϵ-desirable packings that are good approximations of $\textsc{Opt}(\sigma)$. Towards this direction, we need to distinguish between *ϵ-hard* and *ϵ-easy* bins as follows.

Definition 3. *We call a bin* ϵ-hard *if it contains two items of size larger than 1/3 such that the total size of these two items is more than $1-\epsilon$. Otherwise, we call the bin* ϵ-easy.

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

Lemma 8. *Given a set of items packed in m ϵ-easy bins, it is possible to obtain an ϵ-desirable packing of these items using at most $4/3 \cdot m + 2$ bins.*

We now define the packing P_1 and the online algorithm that approximates it. Let H and E denote the number of ϵ-hard and ϵ-easy bins in $\textsc{Opt}(\sigma)$, respectively. Let also γ denote the ratio H/E.

To obtain P_1, we apply the procedure of Lemma 8 to transform the E ϵ-easy bins in $\textsc{Opt}(\sigma)$ into at most $4/3E$ ϵ-desirable bins. Moreover, by applying a procedure similar to Lemma 8, we can transform the H ϵ-hard bins in $\textsc{Opt}(\sigma)$ into at most $1.5H$ ϵ-desirable bins. To summarize, P_1 has at most $1.5H + 4/3E = (1.5\gamma + 4/3)E$ bins (omitting additive constants). From Lemma 7, the DR algorithm outputs a packing with $(1.5\gamma + 4/3 + \epsilon')E$ bins. Comparing this to $\textsc{Opt}(\sigma) = H + E = (1+\gamma)E$, we get the following result.

Lemma 9. *There is an online algorithm that receives advice of constant size (dependant on ϵ) and achieves a competitive ratio of $\frac{9\gamma+8}{6\gamma+6} + \epsilon$.*

Next, we outline the packing P_2 and the online algorithm that approximates it. In particular, we will define the packings P_{2a} and P_{2b} (as described at the beginning of this section). For our analysis, we partition the set of ϵ-easy bins in the optimal packing into four groups depending on the number of items larger than 1/2 or 1/3 in these bins. Let E_1, E_2, E_3 and E_4 indicate the number of bins from these groups ($E_1+E_2+E_3+E_4 = E$). With a similar classifying technique as in Lemmas 8, 9, we obtain P_2 with the following number of bins.

$$\underbrace{H + 1/2 \cdot E_1 + E_2 + E_3}_{P_{2a}:=\text{bins with items}>1/3} + \underbrace{2\epsilon' H + E_1 + 2/3 \cdot E_2 + 4/9 \cdot E_3 + 4/3 \cdot E_4}_{P_{2b}:=\text{desirable bins with items}\leq 1/3}$$

To approximate P_{2a}, since it consists of items of size larger than 1/3, we can use the online algorithm with 1-bit of advice of 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 with 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 10. *There is an online algorithm that receives advice of constant size (dependant on ϵ) and achieves a competitive ratio of $\frac{1.3904\gamma+1.8349}{\gamma+1} + \epsilon$.*

Theorem 3. *There is an online algorithm with advice of constant size (dependant on ϵ) that achieves 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 we apply the algorithm of Lemma 9; this gives a ratio of at most $\frac{9\cdot\gamma^*+8}{6\cdot\gamma^*+6} + \epsilon < 1.470112 + \epsilon$. If $\gamma > \gamma^*$, then we apply the algorithm of Lemma 10; the competitive ratio is at most $\frac{1.3904\cdot\gamma^*+1.8349}{\gamma+1} + \epsilon < 1.47012 + \epsilon$. □

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 [10], which showed a lower bound of 9/8. Both lower bounds use a reduction from a variant of the *binary string guessing problem* (2-SGKH) [12,4]. 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.

In particular, 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). [1]

Lemma 11. *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* $b(n) = (1+(1-\alpha)\log(1-\alpha)+\alpha\log\alpha)n - e(\gamma n) - 1$ *bits of advice, where* $\gamma = \min\{\beta, 1-\beta\}$ *and* $e(\gamma n) = \lceil\log(\gamma n+1)\rceil + 2\lceil\log(\lceil\log(\gamma n+1)\rceil+1)\rceil + 1$.

Given an instance $\mathcal{B}$ of the 2-SGKH$_{1/2}$ problem with a bitstring of length n, we construct a request sequence σ for the online bin packing problem with length $2n$ following [10]. (This is described fully in the long version of the paper.) The sequence consists of a *prefix* of $n/2$ items of size $1/2+\epsilon$, a *central part* of n items of size less than $1/2$ and a *suffix* of $n/2$ items that are the exact complement of the $n/2$ smallest items in the central part. Among these n central items, we refer to the smallest $n/2$ items as *small* items and to the remaining items as *large* items. We observe that $\textsc{Opt}(\sigma) = n$. The $n/2$ small items are packed with their complements in the suffix; moreover, the remaining $n/2$ large items are packed each with an item of the prefix.

Let $\mathbb{B}$ denote an algorithm for the bin packing problem; we will show how to obtain an online algorithm $\mathbb{A}$ for 2-SGKH$_{1/2}$ that constructs σ and uses $\mathbb{B}$. $\mathbb{B}$ must open a bin for the $n/2$ items of the prefix of σ. 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 labeled as small and $\mathbb{A}$ guesses 0. If $\mathbb{B}$ chooses either option (2) or (3), the item is labeled 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 [10], 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 [10], in which the corresponding number equals 4). Let f_n denote the family of all request sequences σ constructed as described above, for all possible bitstrings of length n with exactly $n/2$ 0s.

Lemma 12. *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.*

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* $(1+(1-\alpha)\log(1-\alpha)+\alpha\log\alpha)n - e(n/2) - 1$ *bits of advice to be* ρ*-competitive,* $1 < \rho < 7/6$*, where* $\alpha = 4 - 3\rho$ *and* $e(x) = \lceil\log(x+1)\rceil + 2\lceil\log(\lceil\log(x+1)\rceil+1)\rceil + 1$.

[1] Technically, the statement of Lemma 11 is very similar to Lemma 9 in [10]. 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 11 is parameterized by β, as opposed to Lemma 9 in [10].

References

1. Adamaszek, A., Renault, M.P., Rosén, A., van Stee, R.: Reordering buffer management with advice. In: Proc. 11th International Workshop in Approximation and Online Algorithms (WAOA). pp. 132–143 (2013)
2. Ásgeirsson, E.I., Ayesta, U., Coffman, E.G., Etra, J., Momcilovic, P., Phillips, D.J., Vokhshoori, V., Wang, Z., Wolfe, J.: Closed on-line bin packing. Acta Cybernetica 15(3), 361–367 (2002)
3. Balogh, J., Békési, J., Galambos, G.: New lower bounds for certain classes of bin packing algorithms. Theoretical Computer Science 440–441, 1–13 (2012)
4. Böckenhauer, H., 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. Theoretical Computer Science 554, 95–108 (2014)
5. Böckenhauer, H., Komm, D., Královic, R., Rossmanith, P.: The online knapsack problem: Advice and randomization. Theoretical Computer Science 527, 61–72 (2014)
6. 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). pp. 207–218 (2011)
7. 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). pp. 331–340 (2009)
8. Borodin, A., El-Yaniv, R.: Online Computation and Competitive Analysis. Cambridge University Press (1998)
9. Boyar, J., Kamali, S., Larsen, K.S., López-Ortiz, A.: On the list update problem with advice. In: Proc. 8th International Conf. on Language and Automata Theory and Applications (LATA). pp. 210–221 (2014)
10. Boyar, J., Kamali, S., Larsen, K.S., López-Ortiz, A.: Online bin packing with advice. In: Proc. 31st Symp. on Theoretical Aspects of Computer Science (STACS). pp. 174–186 (2014)
11. Dobrev, S., Královič, R., Pardubská, D.: Measuring the problem-relevant information in input. RAIRO - Theoretical Informatics and Applications 43(3), 585–613 (2009)
12. Emek, Y., Fraigniaud, P., Korman, A., Rosén, A.: Online computation with advice. Theoretical Computer Science 412(24), 2642–2656 (2011)
13. Epstein, L., Levin, A.: On bin packing with conflicts. SIAM J. Optimization 19(3), 1270–1298 (2008)
14. Galambos, G., Woeginger, G.J.: Repacking helps in bounded space online bin packing. Computing 49, 329–338 (1993)
15. Gambosi, G., Postiglione, A., Talamo, M.: Algorithms for the relaxed online bin-packing model. SIAM J. Computing 30(5), 1532–1551 (2000)
16. Grove, E.F.: Online bin packing with lookahead. In: Proc. 6th Symp. on Discrete Algorithms (SODA). pp. 430–436 (1995)
17. Johnson, D.S., Demers, A.J., Ullman, J.D., Garey, M.R., Graham, R.L.: Worst-case performance bounds for simple one-dimensional packing algorithms. SIAM J. Computing 3, 256–278 (1974)
18. Komm, D., Královič, R.: Advice complexity and barely random algorithms. RAIRO - Theoretical Informatics and Applications 45(2), 249–267 (2011)
19. Renault, M.P., Rosén, A.: On online algorithms with advice for the k-server problem. Theory of Computing Systems 56(1), 3–21 (2015)
20. Renault, M.P., Rosén, A., van Stee, R.: Online algorithms with advice for bin packing and scheduling problems. CoRR abs/1311.7589 (2013)
21. Seiden, S.S.: On the online bin packing problem. Journal of the ACM 49, 640–671 (2002)