Almost Online Square Packing

Shahin Kamali Alejandro López-Ortiz *

Abstract

In the square packing problem, the goal is to pack squares of different sizes into the smallest number of bins (squares) of uniform size. We introduce an almost-online square packing algorithm which places squares in an online, sequential manner. In doing so, it receives advice of logarithmic size from an offline oracle which runs in linear time. Our algorithm achieve a competitive ratio of at most 1.84 which is significantly better than the best existing online algorithm which has a competitive ratio of 2.1187. In introducing the algorithm, we have been inspired by the advice model for the analyses of online problems. Our algorithm can also be regarded as a streaming algorithm which packs an input sequence of squares in two passes using a space of logarithmic size.

1 Introduction

In this paper, we consider the square packing problem in which squares of different sizes, called *items*, should be packed into unit squares called *bins*. The problem is a generalization of the classical bin packing problem into two dimensions. We refer to a square by its *size*, which is the length of each of its sides. Each bin has size 1 which is an upper bound for the size of the input squares. Given a set of squares, we would like to place them into the smallest number of bins so that there is no overlap between two squares assigned to a bin.

The problem is studied under both offline and online settings. In the offline setting, all squares are available in advance and the algorithm can look at the whole input for placing the squares. In the online setting, the squares appear in a sequential manner, and for placing a square an algorithm cannot look at future squares. Moreover, the decisions of the algorithm are irrevocable, i.e., it is not possible to move a square from one bin to another after it is placed into a bin.

Square packing problem has many practical applications from storage to cutting stock and other areas [12]. In most cases, the online setting is preferable, partially because it is more representative and the squares naturally appear in an online manner. Moreover, most offline approaches involve solving an integer programming formulation of the problem and are too complicated to be applied in practice. In this paper, we consider an *almost online* setting in which a fast offline oracle provides some useful information to the online algorithm. This information, referred to as *advice*, can significantly boost the performance of the online algorithm. We restrict the offline oracle to run in linear time and only make one pass to collect some basic statistics about the input. Assuming the advice of size $O(\log n)$, this setting of the problem is closely related to the streaming model [1] and map-reduce model [21]. We define the almost-online square packing problem as follows:

definition 1 *In the* almost online square packing problem*, the input is a sequence of squares (items) with sizes $\sigma = \langle x_1, \ldots, x_n \rangle$ which is revealed in an online manner $(0 < x_i \leq 1)$. The goal is to pack these squares into the minimum number of squares of unit size (bins). At time step t, an online algorithm should pack square x_t into a bin. The decision of the algorithm to select the target bin is a function of $\Phi, x_1, \ldots, x_{t-1}$, where Φ is advice of size $O(\log n)$ generated by an offline oracle that runs in linear time (linear in the size of the input).*

The *asymptotic* competitive ratio is the standard method for comparing online (and almost online) algorithms. We say an algorithm $\mathbb{A}$ has a competitive ratio of c if there exists a constant c_0 such that, for all n and for all input sequences σ of length n, we have $\mathbb{A}(\sigma) \leq c \ \textsc{Opt}(\sigma) + c_0$ where $\mathbb{A}(\sigma)$ and $\textsc{Opt}(\sigma)$ denote the costs of $\mathbb{A}$ and Opt for processing σ, respectively, and are both arbitrarily large.

1.1 Related Work

The offline version of the square packing problem is known to be NP-hard [22]. There has been efforts for introducing algorithms with good approximation ratios (see, e.g., [20, 13]). In [2], an APTAS is introduced for the problem, i.e., an algorithm with cost $(1+\epsilon)\,\textsc{Opt}(\sigma) + 1$ for an input σ and a given $\epsilon > 0$.

For the online setting, the first set of results included an upper bound of 2.6875 and a lower bound of 4/3 [10]. The upper bound was improved a few times ([13, 14, 17]). The best existing upper bound is given by an algorithm with a competitive ratio of 2.1187 [18]. In [23], a lower bound of 1.62176 was proved for the competitive ratio of any online algorithm. This lower bound was later improved to 1.6406 [14].

*School of Computer Science, University of Waterloo, .95{s3kamali,alopez-o}@uwaterloo.ca

It should be mentioned that most existing results extend to the *cube packing* problem which asks for packing of cubes of dimension $d \geq 2$. Another generalization of the problem is box packing problem in which there is a sequence of boxes (rectangles) to be packed into unit squares. This generalization is much harder than the square packing. In the offline setting, in contrast to the square packing, there is no APTAS for the box packing problem unless P=NP [2]. The best existing approximation algorithm has an approximation ratio of 1.4055 [3]. It is also known that no algorithm can have an approximation ratio better than $1 + 1/2196$ [9]. The best existing online box packing algorithm has a competitive ratio of 2.5545 [16] while there is a lower bound of 1.907 for the competitive ratio of any online box packing algorithm [4].

The total lack of information about the future is unrealistic in some real-world scenarios [11]. The advice model is introduced to address this issue. Under the advice model, an online algorithm receives a number of bits of advice about the unrevealed parts of the sequence. Generally, there is a trade-off between the size of advice and the performance of online algorithms regarding competitive ratio. The advice model has received significant attention in the past few years (see, e.g., [6, 19, 11, 5, 15, 7]). In particular, the classical bin packing problem is studied under the advice model [8].

1.2 Contribution

We introduce an almost-online algorithm for the square packing problem which achieves a competitive ratio of at most 1.84. The algorithm receives advice of size $\Theta(\log n)$ from an offline oracle that scans the input sequence of length n in linear time. The offline oracle simply counts the number of squares whose sizes lie different intervals defined by the algorithm. The online algorithm uses the advice to pack squares in an efficient way to ensure a good competitive ratio. The algorithm is quite simple and runs as quickly as its online counterparts.

In the context of advice, our algorithm indicates that advice of logarithmic size is sufficient to outperform online algorithms which are blind about future. Note that the competitive ratio of our algorithm is significantly better than 2.1187 of the best existing algorithm [18]. The algorithm can also be regarded as a streaming algorithm with two passes. Although the algorithm does not perform as well as the offline algorithms (in particular the APTAS algorithm), its simple and fast nature makes it useful even in the offline setting. This is particularly because the existing offline algorithms rely on heavy computation, e.g., linear program solvers.

2 Algorithm

In this section we introduce our square packing algorithm. The algorithms define *types* for squares based on their sizes. A square has type i if its size is in the interval $(\frac{1}{i+1}, \frac{1}{i}]$ for $1 \leq i \leq 14$. Squares of size smaller than $1/15$ have type 15 and are referred to as *tiny* squares. Squares of type 1 are called *large* squares and are further divided into types 1a, 1b, 1c, and 1d with sizes in intervals $(4/5, 1]$, $(2/3, 4/5]$, $(3/5, 2/3]$, and $(1/2, 3/5]$, respectively. Similarly, squares of type 2 are divided into types 2a and 2b with sizes in intervals $(2/5, 1/2]$ and $(1/3, 2/5]$, respectively. We refer to items of type 2 as *medium* items and items of types $3, 4, \ldots, 14$ as *small* items.

In total, there are a constant number of item types. The offline oracle simply scans the input and counts the number of items for each type except the last type associated with the tiny items. These numbers are encoded as advice of size $\Theta(\log n)$. The online algorithm makes use of this advice to achieve a competitive ratio of at most 1.84. To describe the algorithm, we start with the following two lemmas (note the distinction between the *size* and the *area* of a square).

Lemma 1 *[13] Consider the square packing problem in which all items are smaller than or equal to $1/M$ for some integer $M \geq 2$. There is an online algorithm that creates a packing in which all bins, except possibly one, have an occupied area of size at least $(M^2-1)/(M+1)^2$.*

Lemma 2 *There is an online square packing which creates packings in which all bins, except possibly a constant number of them, have an occupied area of size more than $1/4$.*

Proof. Consider an online algorithm that places each large square in a separate bin. Since these items have size large than $1/2$, the occupied area in each bin is more than $1/4$. For squares in interval $(1/3, 1/2]$, the algorithm places four squares in the same bin; the total occupied area will at least $4/9 > 1/4$. For squares that are no larger than $1/3$, the same algorithm of Lemma 1 is applied which ensures that the total volume of each bin is at least $1/2$. □

Given the advice, the online algorithm knows upper and lower bounds for the size of all items (except tiny items). Before serving the sequence in an online manner, the algorithm creates an *approximate packing* in which there is a *reserved* area for each non-tiny item. The reserved area for an item x is equal to the upper-bound for the actual area of the square (which is revealed later). To create the approximate packing, the algorithm opens a new bin for each large item. Note that no two large squares can fit into one bin. Moreover, the algorithm opens a new bin for each four squares of type

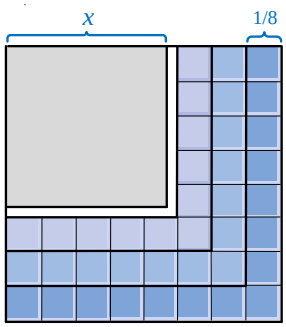


Figure 1: L-shape tiling for small items of type 8.

2. In doing so, it treats squares of type 2a and 2b separately, i.e., it does not place a 2a item and a 2b item in the same bin.

Depending on the type of the large and medium items, there might be enough space for small items in the opened bins. Assume a square of size x is reserved for a large item or a group of medium items in a bin B. Also, assume the algorithm places x on the top-left corner of the bin. We use *L-shape tiling* to place small items of the same types into B. As before, by 'placing' a small item we mean reserving a sufficient space for the item in the approximate packing. The size of items of type $i \geq 3$ is in interval $(1/(i+1), 1/i]$. Hence, $2i-1$ items of this type can be placed on the right and bottom sides of the bin; these squares form an L shape set of tiles. In case $1 - x > 1/i$, there is enough space for L-shape tiling of another $2i-3$ items of type i (see Figure 1). More generally, we prove the following lemma.

Lemma 3 *Given a bin in which an area for a square of size at most $x \geq 1/2$ is reserved, one can apply the L-shape tiling to place $2ki - k^2$ small items of type $i \in \{3, 4, \ldots, 14\}$ into the bin where $k = \lfloor (1-x)i \rfloor$.*

Proof. Since the small items are in range $(1/(i+1), 1/i]$, the first L-shape includes $2i-1$ items. The second L-shape includes one less item on each side and includes $2i-3$ items. More generally, the jth L-shape includes $2i + (2j-1)$ items. The total number of items after k iterations of L-shape tiling will be $\sum_{j=1}^{k} 2i + (2j-1) = 2ki - k^2$. Moreover, k items of type i require a width of k/i and we should have $k/i + x \leq 1$ which implies $k \leq (1-x) \times i$. □

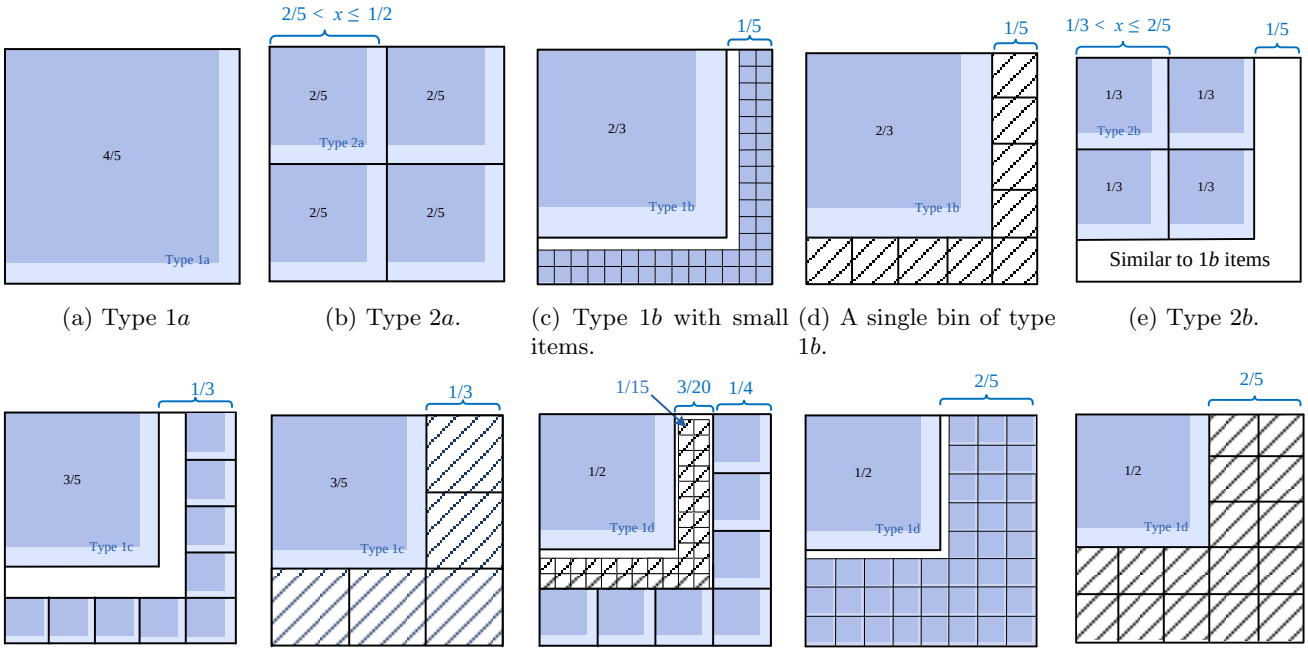
As mentioned earlier, the algorithm opens a new bin for each large square as well as each four medium squares. The bins opened so far are called LM-bins (Large-Medium bins). Initially, all LM-bins are *single*, i.e., there is no reserved area for small items in them. The algorithm uses L-shape tiling to place small items into the LM-bins (in the approximate packing) as follows. An LM-bin with a square of type 1a or four squares of type 2a remains single, i.e., no other items will be placed there (Figures 2a,2b). An LM-bin with a large item of type 1b has enough space for small items of types 5 or larger and the algorithm applies L-shape tiling to fill the empty space with these items (Figures 2c). If there are not enough small items, some of these bins might remain single (Figure 2d). Any four items of type 2b form a single square whose size is in interval $(2/3, 4/5]$ and will be treated like a 1b item, i.e., the algorithm places small items of types 5 or larger in these bins (Figure 2e).

In the same manner, the algorithm applies L-shape tiling to fill the available area in the bins having an item of type 1c and 1d with small items of types 3 or larger (Figures 2f,2i). Again, if there are not enough small items, some bins will remain single (Figure 2g,2j).

To summarize, we use L-tiling to place small items in the opened LM-bins. In case there are not enough LM-bins for placing small items, we open new bins for them. We call these bins *harmonic bins*. A harmonic bin of type $i \geq 3$ includes i^2 small items of type i. LM-bins and harmonic bins form the approximate packing of an input sequence.

Inside some LM-bins, there is an empty area which will be used for placing tiny items. If there is a single LM-bin in the approximate packing, all the available space (i.e., the area which is not reserved for large or medium items) can be partitioned into squares of size $1/5$ or larger. We call these squares *large live squares*. Moreover, for LM-bins with 1d items accompanied by small items of type 4, the empty area can be used to reserve 40 squares of sizes $1/15$ (figures 2h). We refer to these squares as *small live squares*.

Provided with the approximate packing, the online algorithm places items in the input sequence in the following manner. Any non-tiny item is placed in the reserved area for its type in the approximate packing. To place tiny items, the algorithm first uses the live squares. Note that the size of live squares are at least $1/15$ which is the upper bound for the size of tiny items. Large live squares have size at least $1/5$; hence we can use Lemma 1 to place tiny items in the large squares so that at least half of their area be occupied by tiny items. We declare a large live square as closed if half of its area is occupied. If all large live squares are closed, we use the algorithm of Lemma 2 to place tiny items in the small live squares. We declare a small live square as closed if its occupied area is at least a quarter of its total area. If all large and small live squares are closed, a new bin is opened for the tiny items and again the algorithm of Lemma 1 is applied. We call these bins *tiny* bins. This way, there are possibly three types of bins in the final packing of the algorithm, namely, LM-bins, Harmonic bins, and tiny bins.



(a) Type 1a (b) Type 2a. (c) Type 1b with small items. (d) A single bin of type 1b. (e) Type 2b.

(f) Type 1c with small items. (g) A single bin of type 1c. (h) Type 1d with small items of type 4. (i) Type 1d with small items of type $\neq 4$. (j) A single bin of type 1d.

Figure 2: A summary of LM-bins in an approximate packing. The dark squares indicate the lower bound for the size of the squares of different types while the light parts indicate the upper bound (i.e., the reserved space). The striped squares indicate the live squares which are used for the tiny items.

2.1 Analysis

In this section, we prove that our algorithm has a competitive ratio of at most 1.84. Consider an LM-bin in the approximate packing which has a reserved area of size x for large or medium items, while the remaining area is filled with (reserved for) small items of type i. Let m denote the number of small squares in the bin given by Lemma 3. The occupied area by these squares in the final packing is at least $m \times 1/(i+1)^2$ since the size of a small bin of type i is more than $1/(i+1)^2$. Table 1 indicates the minimum area covered area by small items of type i in LM-bins which include large or medium items of type j. For example, when $j = 1c$ and $i = 4$, the reserved space for the large item is $2/3$ and by lemma 3, 7 items of type 4 can be placed in the bin. These items occupy an area of size at least $7 \times 1/25$ as indicated in the table. In our analysis, we are interested in the minimum covered area by all small items in an LM-bin, i.e., the minimum value in the rows of Table 1. For example, for LM-bins with a large item of type 1c, the minimum occupied area is given by small items of type 5 where their total occupied area is at least $1/4$. Table 1 also indicates the occupied area by tiny items when they are placed in the 'large' live squares of single bins. For example, for a single bin in the approximate packing which contains a 1c item, there are 5 live squares with total area of $5/9$; at least half of this area (i.e., $5/18$) is occupied by the tiny items, as indicated in Table 1.

To prove the upper bound for the competitive ratio, we consider a few cases separately in the following lemmas.

Lemma 4 *Assume there is a tiny bin in the final packing of the algorithm. Then the occupied area in all bins, except possibly a constant number of them, is more than* $9/16$.

Proof. We prove the claim for tiny, harmonic, and LM-bins separately. Tiny bins are opened using the algorithm of Lemma 1; since the size of items is smaller than $1/15$, the occupied area of all bins, except possibly one of them, is at least $224/256 > 9/16$. A harmonic bin of type i has place for i^2 items of type i ($i \geq 3$). Hence, the occupied area of such a bin is more than $i^2/(i+1)^2$. This value increases as i grows which implies that the minimum occupied area of a harmonic bin is $9/16$.

Next, we consider LM-bins. The occupied area of bins which include an item of type 1a or four items of type 2a is at least $16/25 > 9/16$. We know that other LM-bins cannot be single; otherwise, the algorithm would have placed tiny items in those bins (instead of opening new bins).

Assume a bin includes an item of type 1b or four

Type j	Reserved	$i=3$	$i=4$	$i=5$	$i=6$	$i=7$	$i=8$	$i=9$	$i=10$	$i=11$	$i=12$	$i=13$	$i=14$	tiny
1a,2a	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1b,2b	4/5	0	0	1/4	11/49	13/64	15/81	**17/100**	36/121	40/144	44/169	48/196	52/225	9/50
1c	2/3	5/16	7/25	**1/4**	20/49	24/64	28/81	45/100	51/121	57/144	80/169	88/196	96/225	5/18
1d	3/5	5/16	**7/25**	16/36	20/49	24/64	39/81	45/100	64/121	72/144	80/169	105/196	115/225	16/50

Table 1: Lower bounds for the occupied area by the small items of type i in the LM-bins. The last column indicate the occupied area by the tiny items when there is no small item in the bin (i.e., the bin is single in the approximate packing). The highlighted numbers indicate the minimum covered area among all types of small and tiny items.

items of type 2b. Since the bin is not single, it is either accompanied by small or tiny items. In both cases, as Table 1 suggests, the occupied area by small or tiny items is at least 17/100. Hence, the occupied area of the bin is more than $4/9 + 17/100 > 9/16$. With a similar argument, the occupied area of bins with a 1c item is more than $9/25 + 1/4 > 9/16$.

Consider an LM-bin which includes a 1d item. If the bin includes small items of type $i \neq 4$ or tiny items, as Table 1 indicates, the small or tiny items occupy an area of size at least 16/50 and the occupied area of the bin will be more than $1/4 + 16/50 > 9/16$. If the bin includes small items of type 4, there are 40 live small squares in the bin (Figure 2h). Since the algorithm has opened tiny bins, by Lemma 2, at least 1/4 of the total area of the live squares is occupied. Also, by Table 1, the total occupied area of the small items is at least 7/25. In total, the occupied area of the bin is more than $1/4 + 7/25 + 1/4 \times 40/225 = 517/900 > 9/16$. □

The above lemma implies that if the algorithm opens a tiny bin, the number of opened bins by the algorithm is at most $Ar \times 16/9$ where Ar is total area of all items in the sequence. Since Opt opens at least Ar bins, the competitive ratio of the algorithm at most $16/9 < 1.84$.

Lemma 5 *Assume there is no tiny bin while there is a harmonic bin of type $i \geq 5$ in the final packing of the algorithm. Then the competitive ratio is no more than* 1.84.

Proof. Similar to many upper-bound arguments for packing algorithms, we use a weighting function as follows. We define the weight of items of types 1a and 2a to be respectively 1 and 1/4. Tiny items have weight 0.

Type	Size	Area	Weight	Density
1a	$x \in (4/5, 1]$	$x^2 > 16/25$	1	< 1.5625
1b	$x \in (2/3, 4/5]$	$x^2 > 4/9$	$1 - 0.17\alpha \approx 0.698$	< 1.57
1c	$x \in (3/5, 2/3]$	$x^2 > 9/25$	$1 - 0.25\alpha \approx 0.556$	< 1.55
1d	$x \in (1/2, 3/5]$	$x^2 > 1/4$	$1 - 7\alpha/25 \approx 0.503$	< 2.01
2a	$x \in (2/5, 1/2]$	$x^2 > 4/25$	0.25	< 1.5625
2b	$x \in (1/3, 2/5]$	$x^2 > 1/9$	$0.25 - 0.0425\alpha \approx 0.1745$	< 1.57
$3, \ldots, 14$	$x \in (1/15, 1/3]$	-	αx^2	$\alpha \approx 1.78$
15	$x \in (0, 1/15]$	-	0	0

Table 2: Characteristics of items of different types in Lemma 5.

Let α be a constant equal to 16/9. A small item of size x and of type 3 or larger has weight αx^2. The weight of any other item y is defined as $1 - w$ where w is a lower bound for the total weight of small or tiny items accompanied with y in the bin. Note that, since there is a harmonic bin of type ≥ 5, large and medium items like y are accompanied with some small or tiny items.

For bins with an item of type 1b, the minimum area occupied by small or tiny items is at least 17/100 (see Table 1). So, the weights items of type 1b is $1 - 0.17\alpha \approx 0.698$. Similarly, the weight of items of types 1c and 1d are defined as $1 - \alpha/4 \approx 0.556$ and $1 - 7\alpha/25 \approx 0.503$, respectively. Items of type 2b have weight $1/4 - 17\alpha/400 < 0.1745$ which is the same as a 1b item when divided between four 2b items placed in the bin.

In our analysis, we refer to *density* of an item as the ratio between its size and area. Table 2 provides a summary of the weights and densities of items. To prove the lemma, we show the followings:

claim1: All bins in the final packing, except possibly a constant number of them, have weight at least 1.

claim2: It is not possible to place a set of items into a bin so that the total weight of the items exceeds 1.84.

Claim 1 implies that the cost of the algorithm is at most equal to W, which is the total weight of items in the sequence. Claim 2 implies that the cost of Opt is at least $W/1.84$. Hence, the two claims together prove the lemma.

For claim 1, note that there is no tiny bin in the final packing of the algorithm. The bins which include a 1a or four 2a items clearly have weight 1. For all other LM-bins, the weights of the large and medium items are defined in a way to ensure that the accumulated weight of the bin is no less than 1 (except possibly a constant number of bins). A harmonic bin of type $i \geq 3$ includes i^2 items of type i; these items occupy an area of size at least $1/(i+1)^2$. The total occupied area in these bins is then $i^2/(i+1)^2 \geq 9/16$. Hence, the total weight of these items is at least $9/16 \times \alpha = 1$.

For claim 2, note that if there is no item of type 1d in the bin, the density of all items will be less than 1.84 and consequently their total weight is no more than 1.84. Assume there is an item of type 1d with weight $1 - 0.28\alpha$. There is an available area of size less than 3/4 for other items. The maximum density of items in in this area is α. Consequently, the total weight of items in the bin is

Type	Size	Area	Weight	Density
1a	$x \in (4/5, 1]$	$x^2 > 16/25$	1	< 1.5625
1b	$x \in (2/3, 4/5]$	$x^2 > 4/9$	1	< 2.25
1c	$x \in (3/5, 2/3]$	$x^2 > 9/25$	9/16	< 1.5625
1d	$x \in (1/2, 3/5]$	$x^2 > 1/4$	9/16	< 2.25
2a	$x \in (2/5, 1/2]$	$x^2 > 4/25$	0.25	< 1.5625
2b	$x \in (1/3, 2/5]$	$x^2 > 1/9$	0.25	< 2.25
3	$x \in (1/4, 1/3]$	$> 1/16$	1/9	< 1.78
4	$x \in (1/5, 1/4]$	$> 1/25$	1/16	< 1.5625
$5, \ldots, 15$	$x \in (0, 1/15]$	-	0	0

Table 3: Characteristics of items of different types in Lemma 6.

less than $1 - 0.28\alpha + 0.75\alpha = 1 + 0.47\alpha < 1.84$ □

Using a similar approach, we prove the following two lemmas.

Lemma 6 *Assume there is no tiny bin or Harmonic bin of type ≥ 5 in the final packing of the algorithm while there is a harmonic bin of type $i \in \{4, 5\}$. Then the competitive ratio is no more than* 1.84.

Proof. We use a weighting argument as before. The weights of items of types 1a and 1b are 1 while the weight of items of type 2 is 1/4 and the weights of items of type 3 and 4 are respectively 1/9 and 1/16. The weight of all tiny items and small items of types ≥ 5 is 0. For the items of types 1c and 1d, we consider the minimum weight of small items accompanied with them in the same bins. Note that 1c and 1d items cannot be single since some harmonic bins of type 3 or 4 are opened. The contributed weight by small items is at least $\min(5 \times 1/9, 7 \times 1/16) = 7/16$. Hence, we define the weights of 1c and 1d items to be 9/16. This way, the weights of all bins in the final packing of the algorithm is at least 1 (see Table 3 for a summary of weights and densities).

Next, we show that no bin in the packing of Opt can have weight more than 1.84. Assume there is a bin with weight more than 1.84. As entries in Table 3 indicate, the bin should contain items of type 1b, 1d, or 2b; otherwise, the density of all items and consequently the weight of the bin will be less than 1.84. First, assume there is no item of type 1d in the bin. Note that two items of types 1b and 2b do not fit in the same bin (they have sizes more than 2/3 and 1/3, respectively). Only 1 item of type 1b or four items of type 2b fit in the same bin. In both cases, the contributed weight of non-small items is 1. Moreover, these items occupy an area of size more than 4/9; hence, there is enough space for at most 5 items of type 3 and 2 items of type 4 (Figure 3a), and the weight of the bin will be $1 + 5 \times 1/9 + 2 \times 1/16 \approx 1.68 < 1.84$. Next, assume there is an item of type 1d in the bin. Intuitively speaking, to achieve the maximum total weight, we need to fill the bin with items of high density; however, this results in a lot of empty space (since items with high density are large and do not fit each other in a bin). To be more precise, if there are 3 items of type 2b in the bin, then there is space for at most 2 items of type 3 and 2 items of type 4, and the total weight will be $9/16 + 3 \times 0.25 + 2 \times 1/9 + 2 \times 1/16 < 1.84$ (Figure 3b). Similarly, when there are respectively 2 or 1 items of type 2b in the bin, the total weight of the bin will be at most $9/16 + 2 \times 0.25 + 3 \times 1/9 + 4 \times 1/16 < 1.84$ (Figure 3c) and $9/16 + 0.25 + 4 \times 1/9 + 6 \times 1/16 < 1.84$ (Figure 3d). If there is no item of type 2b in the bin, there can be at most 5 items of type 3 and 7 items of type 4 (Figure 3e). The total weight of the bin will be $9/16 + 5 \times 1/9 + 7 \times 1/16 < 1.84$. □

Lemma 7 *Assume there are no tiny or harmonic bins in the final packing of the algorithm. Then the competitive ratio is no more than* 1.75.

Proof. We define the weight of items of types 1 and 2 to be respectively 1 and 1/4, while the weights of all other items (i.e., small and tiny items) are 0. Since there is no harmonic or tiny bin, the weights of all bins is 1. Also, no bin in the offline packing has weight more than 1.75. This is because if a bin contains an item of type 1 (size more than 1/2), then it cannot contain more than 3 items of type 2 (size more than 1/3); in this case, the total weight of items is $1 + 3 \times 1/4 = 1.75$. Note that a bin that only contains items of one type (1 or 2) has weight 1. □

From Lemmas 4, 5,6, and 7, we conclude the following theorem:

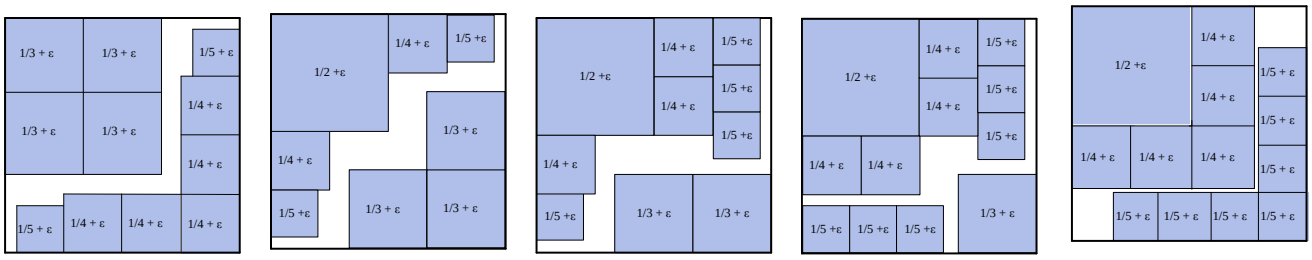
Theorem 8 *There is an almost online algorithm for the square packing problem which receives advice of size $\Theta(\log n)$ for a sequence of size n and achieves a competitive ratio of at most* 1.84.

3 Remarks

The algorithm introduced in this paper is expected to be generalized to the cube packing problem with d-dimensional cubes ($d \geq 2$). Providing almost-online algorithms for the box packing problems seems more challenging and we leave it as a future work. Another promising direction is to investigate how many bits of advice are required and sufficient to achieve a 1-competitive algorithm for the square packing problem.

References

[1] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Systems Sci.*, 58(1):137–147, 1999.



(a) There are items of type 1b or 2b.

(b) An item of type 1d accompanied by three items of type 2b.

(c) An item of type 1d accompanied by two items of type 2b.

(d) An item of type 1d accompanied by an item of type 2b.

(e) There is an item of type 1d and no item of type 1b.

Figure 3: Packings which result in the maximum total weight in a bin of Opt in different cases.

[2] N. Bansal, J. R. Correa, C. Kenyon, and M. Sviridenko. Bin packing in multiple dimensions: Inapproximability results and approximation schemes. *Math. Oper. Res.*, 31(1):31–49, 2006.

[3] N. Bansal and A. Khan. Improved approximation algorithm for two-dimensional bin packing. In *Proc. 25th Symp. on Discrete Algorithms (SODA)*, pages 13–25, 2014.

[4] D. Blitz, A. van Vliet, and G. J. Woeginger. Lower bounds on the asymptotic worst-case ratio of online bin packing algorithms. Unpublished manuscript, 1996.

[5] H.-J. Böckenhauer, D. Komm, R. Královič, and R. Královič. On the advice complexity of the k-server problem. In *Proc. 38th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 6755 of *Lecture Notes in Comput. Sci., Springer*, pages 207–218, 2011.

[6] H.-J. Böckenhauer, D. Komm, R. Královič, R. Královič, and T. Mömke. On the advice complexity of online problems. In *Proc. 20th International Symp. on Algorithms and Computation (ISAAC)*, volume 5878 of *Lecture Notes in Comput. Sci., Springer*, pages 331–340, 2009.

[7] J. Boyar, S. Kamali, K. S. Larsen, and A. López-Ortiz. On the list update problem with advice. In *Proc. 8th International Conf. on Language and Automata Theory and Applications (LATA)*, pages 210–221, 2014.

[8] J. Boyar, S. Kamali, K. S. Larsen, and A. López-Ortiz. Online bin packing with advice. In *Proc. 31st Symp. on Theoretical Aspects of Computer Science (STACS)*, pages 174–186, 2014.

[9] M. Chlebík and J. Chlebíková. Inapproximability results for orthogonal rectangle packing problems with rotations. In *Proc.65th International Conf. on Algorithms and Complexity (CIAC)*, pages 199–210, 2006.

[10] D. Coppersmith and P. Raghavan. Multidimensional online bin packing: Algorithms and worst case analysis. *Oper. Res. Lett.*, 8:17–20, 1989.

[11] Y. Emek, P. Fraigniaud, A. Korman, and A. Rosén. Online computation with advice. *Theoret. Comput. Sci.*, 412(24):2642 – 2656, 2011.

[12] L. Epstein and A. Levin. Robust approximation schemes for cube packing. *SIAM J. Optim.*, 23(2):1310–1343, 2013.

[13] L. Epstein and R. van Stee. Optimal online bounded space multidimensional packing. In *Proc. 15th Symp. on Discrete Algorithms (SODA)*, pages 214–223, 2004.

[14] L. Epstein and R. van Stee. Online square and cube packing. *Acta Inform.*, 41(9):595–606, 2005.

[15] M. Forišek, L. Keller, and M. Steinová. Advice complexity of online coloring for paths. In *Proc. 6th International Conf. on Language and Automata Theory and Applications (LATA)*, volume 7183 of *Lecture Notes in Comput. Sci., Springer*, pages 228–239, 2012.

[16] X. Han, F. Y. L. Chin, H.-F. Ting, G. Zhang, and Y. Zhang. A new upper bound 2.5545 on 2d online bin packing. *ACM Trans. Algorithms*, 7(4):1–18, Sept. 2011.

[17] X. Han, D. Ye, and Y. Zhou. Improved online hypercube packing. In *Proc. 4th International Workshop in Approximation and Online Algorithms (WAOA)*, pages 226–239, 2006.

[18] X. Han, D. Ye, and Y. Zhou. A note on online hypercube packing. *Cent. Eur. J. Oper. Res.*, 18(2):221–239, 2010.

[19] J. Hromkovič, R. Královič, and R. Královič. Information complexity of online problems. In *Proc. 35th Symp. on Mathematical Foundations of Computer Science (MFCS)*, volume 6281 of *Lecture Notes in Comput. Sci., Springer*, pages 24–36, 2010.

[20] Y. Kohayakawa, F. K. Miyazawa, P. Raghavan, and Y. Wakabayashi. Multidimensional cube packing. *Elect. Notes on Discrete Math.*, 7:110–113, 2001.

[21] R. Lämmel. Google's mapreduce programming model - revisited. *Sci. Comput. Program.*, 70(1):1–30, 2008.

[22] J. Y. T. Leung, T. W. Tam, C. S. Wong, G. H. Young, and F. Y. L. Chin. Packing squares into a square. *J. Parallel Distrib. Comput.*, 10(3):271–275, 1990.

[23] S. S. Seiden and R. van Stee. New bounds for multidimensional packing. *Algorithmica*, 36(3), 2003.