Online Square Packing with Rotation

Shahin Kamali* Pooya Nikbakht†

Abstract

We consider the square packing problem, where the goal is to place a multiset of square items of different side-lengths in $(0, 1]$ into a minimum number of square bins of uniform side-length 1. We study the problem under the online setting, where the multiset of items forms a sequence revealed in an online and sequential manner. An online algorithm must place each item into a square bin without prior knowledge of the forthcoming items. Most existing results assume square items are placed orthogonally to the square bins (that is, parallel to the sides of the bins). In the presence of rotation, Kamali and Nikbakht [COCOA 2020] proved that the offline problem is NP-hard and admits an APTAS in an augmented setting. This paper investigates the online problem when item rotation is allowed. We introduce a linear-time algorithm that achieves an asymptotic competitive ratio of 2.306 when square-items have any size $x \in (0, 1]$, and a better asymptotic competitive ratio of 1.732 when $x \in (0, 1/2]$. We also study another problem where items, instead of squares, are isosceles right triangles (half-squares) and present a linear-time online algorithm with an asymptotic competitive ratio of at most 1.897.

1 Introduction

An instance of the *square packing problem* is defined with a multiset of *squares-items* of different sizes in the range $(0, 1]$. The goal is to place these squares into a minimum number of unit *square-bins* in a way that two square items placed in the same square bin do not intersect. At the same time, they can still "touch" each other. The problem is a generalization of the classical bin packing problem into two dimensions, and we sometimes refer to the squares-items simply as "items" and square-bins as "bins." A square item can be recognized by its side-length, which we refer to as the *size* of the square.

In the offline setting, all square-items are given in advance, and the algorithm can process them as a whole before placing any item into a bin. In particular, the algorithm can sort squares in decreasing order of their sizes, which comes in handy in designing algorithms. In the online setting, the multi-set of items forms a *sequence* which is revealed online and sequentially. When an item is revealed, an online algorithm must place it into a square bin without prior knowledge of forthcoming items. The decisions of an online algorithm are irrevocable.

Square packing has many applications in practice. One application is cutting stock where bins represent stocks (e.g., wood boards) and items are requests to squares of specific sizes. When requests arrive, an algorithm must cut the stock to provide the pieces that match the requests. This cutting process is equivalent to placing items into bins. Note that cutting stock aims to minimize the number of stocks, which also matches a square packing goal. We note that in many practical applications, requests arrive in an online manner, and the stock should be cut without prior knowledge about future requests. It is needless to say that the cutting process is irrevocable, which gives an inherently online nature to these applications of square packing.

There has been a rich body of research around square packing. All existing results except for recent work on the offline problem by Kamali and Nikbakht [20] assume that squares are not allowed to rotate; that is, the sides of square-items should be parallel to the square-bins. While this assumption makes the combinatorial analysis of the problem more straightforward, it comes at a cost. For example, consider an instance of the problem formed by n items of size 0.36. If we do not allow rotation, any bin can include at most four items, giving any algorithm a total cost of $n/4$. Allowing rotation, however, five items fit in each bin, and we can reduce the cost to $n/5$ (see Figure 1). As a result, the number of required bins is decreased by $n/20$, which is a notable saving in practice, e.g., for cutting stock applications.

In [20], it was proved that the offline problem is NP-

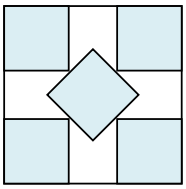


Figure 1: If all items have a length 0.36, allowing rotation helps pack 5 items per bin instead of 4.

*Department of Electrical Engineering and Computer Science, York University, Toronto, Canada, kamalis@yorku.ca

†Department of Computer Science, University of Manitoba, Winnipeg, Canada, nikbakhp@myumanitoba.ca

complete in the presence of rotation and admits an APTAS in an augmented setting, where the bins of the online algorithm are slightly larger than those of the optimal offline algorithm. In this paper, we consider the online square packing problem with rotation:

Definition 1 *In the online square packing with rotation, the input is a sequence $\sigma = \langle a_1, a_2, \ldots, a_n \rangle$ which is revealed in an online manner. At time-step t, the value of a_t is revealed, and an online algorithm has to place a square of size a_t into a bin, using any degree of transition and rotation, such that no two items in the same bin intersect (they can touch). The algorithm's decisions are irrevocable and are made without knowing the values of $a_{t'}$ for $t' > t$. The goal is to pack all square bins into a minimum number of squares of unit size.*

The asymptotic competitive ratio is the standard method for analyzing online packing problems. An algorithm A has a competitive ratio of c if there exists a constant $c_0 \geq 0$ such that, for all n and for all input sequences σ of length n, we have $A(\sigma) \leq c \cdot \textsc{Opt}(\sigma) + c_0$ where $A(\sigma)$ and $\textsc{Opt}(\sigma)$ denote the costs of A and an optimal offline algorithm Opt (with unbounded computational power) for processing σ, respectively.

1.1 Related Work

The 1-dimensional bin packing has been studied extensively in offline and online settings (e.g., [13, 12, 6, 7, 18, 1]). In the 1-dimensional setting, each item has a size in $(0, 1]$, and each bin has a capacity of 1. The offline, 1-dimensional bin packing problem is NP-hard [13], and the best existing result is an algorithm that opens at most $\textsc{Opt}(\sigma) + O(\log \textsc{Opt}(\sigma))$ bins σ [16]. In the online setting, the best existing algorithm has a competitive ratio of 1.578 [2], while no online algorithm has a competitive ratio better than 1.54278 [3].

There are many ways to extend bin packing to higher dimensions (see [5] for a survey). Packing axis-aligned square items into square bins is perhaps the most straightforward extension. In the offline setting, the problem is NP-hard [21], and there exists an APTAS for the problem [4]. In the online setting, the upper and lower bounds have been improved a few times [22, 10]. The best existing algorithm has a competitive ratio of 2.1187 [14] while the best existing lower bound is 1.6707 [15]. Almost-online square (and triangle) packing, where an online algorithm receives some "advice" about the input sequence, is studied in [17, 19].

Another generalization of bin packing into two dimensions assumes items are axis-aligned rectangles. This problem is also studied extensively (see [5] for details). In particular, a variant of this problem assumes rectangles can be rotated by exactly ninety degrees (see, e.g., [8]). We note that rotation by ninety degrees is not relevant for square packing and is quite restrictive compared to the rotations considered in this paper.

1.2 Contribution

We study the online square packing problem and present an online algorithm that achieves a competitive ratio of 2.306 for the square packing problem. Our algorithm is based on classifying squares based on their sizes and placing squares of similar sizes tightly, possibly using rotations, in the same bins. This approach was previously used to introduce different families of Harmonic algorithms for the classic bin packing in both one dimension and higher dimensions. However, the presence of rotations makes our classification and analysis different from the previous work. While analyzing the algorithm, we also consider sequences where the item sizes are at most $1/2$ and show that our algorithm has a competitive ratio of at most 1.732 in this case. We also study another problem where items, instead of squares, are isosceles right triangles (half-squares), also called "tans", and present a linear-time online algorithm with an asymptotic competitive ratio of at most 1.897.

2 Online Square Packing

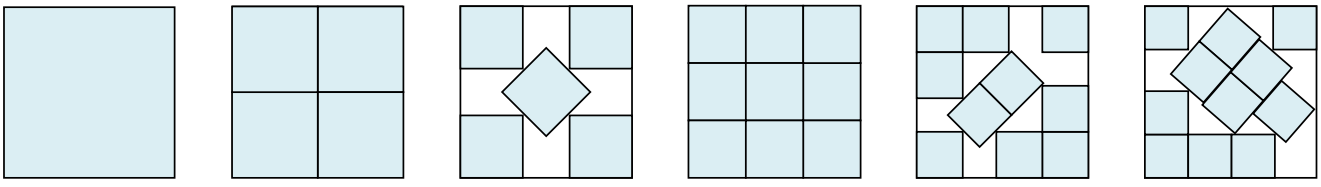
In this section we introduce our square packing algorithm called Square-Rotate.

2.1 Item Classification

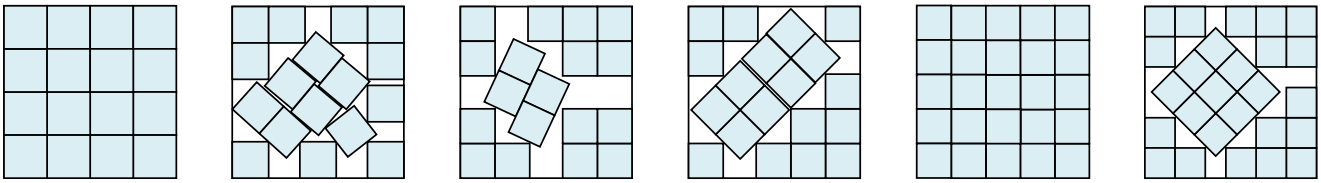
We classify squares by their side lengths, which we refer to as the "size" of the items. Square-Rotate packs squares of each class separately from other classes.

In total, there are 13 classes of squares. Square items with sizes in the range $(0, 0.1752]$ are in class 13; we refer to this class as the tiny class, and items that belong to it are called tiny items. We refer to items that belong to class $i \in [1, 12]$ as regular items. For each class $i \in [1, 12]$, the range of items in the class is specified as $(x_i, x_{i-1}]$ (for convenience, we define $x_0 = 1$). The values of x_i's are defined so that a certain number of items, denoted by S_i, of class i can fit in the same bin. The specific range of item sizes for each class $i \in [1, 12]$ and values of S_i are derived from the best-known or optimal results on the *congruent square packing problem* [11]. This problem asks for a square's minimum size $c(j)$ that can contain j unit-sized squares. A scaling argument, where the container size is fixed to be 1, gives values of $u(j)$'s when the goal is to pack j identical squares of maximum size $u(j)$ into a unit square. Table 1 provides the scaled best-known/optimal $u(j)$ values for $1 \leq j \leq 36$. These scaled numbers give the specific ranges that we use for classifying items.

Items of class 1 have sizes in the range $(1/2, 1]$, and we have $x_1 = 1/2$. Note that exactly $S_1 = 1$ item of



(a) Class 1: $x \in (0.5000, 1.0000]$ (b) Class 2: $x \in (0.3694, 0.5000]$ (c) Class 3: $x \in (0.3333, 0.3694]$ (d) Class 4: $x \in (0.2697, 0.3333]$ (e) Class 5: $x \in (0.2579, 0.2697]$ (f) Class 6: $x \in (0.2500, 0.2579]$



(g) Class 7: $x \in (0.2139, 0.2500]$ (h) Class 8: $x \in (0.2073, 0.2139]$ (i) Class 9: $x \in (0.2047, 0.2073]$ (j) Class 10: $x \in (0.2000, 0.2047]$ (k) Class 11: $x \in (0.1779, 0.2000]$ (l) Class 12: $x \in (0.1752, 0.1779]$

Figure 2: Placement of regular square items of class $i \in [1, 12]$ in their respective bin. It is possible to pack i square items of class i into a single square bin [11].

j	$u(j)$	opt./b.k.	j	$u(j)$	opt./b.k.
1	1.0	opt.	19	0.2047	b.k.
2-4	0.5	opt.	20-22	$= 0.2$	b.k.
5	0.3694	opt.	23-25	$= 0.2$	opt.
6-9	0.3333	opt.	26	0.1779	b.k.
10	0.2697	opt.	27	0.1752	b.k.
11	0.2579	b.k.	28	0.1716	b.k.
12-13	$= 0.25$	b.k.	29	0.1685	b.k.
14-16	$= 0.25$	opt.	30-33	≈ 0.1667	b.k.
17	≈ 0.2139	b.k.	34-36	≈ 0.1667	opt.
18	≈ 0.2073	b.k.			

Table 1: Optimal (opt.) or best-known (b.k.) values of $u(j)$ for $1 \leq j \leq 36$ when the goal is to pack j identical squares of the largest size $u(j)$ into a unit square. Values of $u(j)$ are the scaled values of the known results on congruent square packing [11].

class 1 can fit in the same bin. For $i \in [2, 12]$, S_i is the largest number of items of size x_{i-1} that fit in the same bin. For example, for $i = 2$, we have $S_2 = 4$ because $x_1 = 1/2$, and at most 4 items of size $1/2$ fit in the same bin. Moreover, x_i is defined as the largest value so that $S_i + 1$ items of size x_i do not fit in the same bin. For example, we have $x_2 = 0.3694$ because, according to Table 1, $S_2 + 1 = 5$ squares of size larger than 0.3694 do not fit in the same bin (with respect to the best known results).

The respective range of items for each class, as well as the values of S_i, is presented in Table 2. For example, a square is in class 1, 2, or 12 if its side size is in the interval $(0.5, 1]$, $(0.3694, .5]$, or $(0.1752, 0.1779]$, respectively. In Figure 2, it is specified how S_i items of the largest size in class i can fit into a square bin. We refer to [11] for details on the unit square packing problem.

2.2 Packing Regular Items

For each class i ($1 \leq i \leq 12$), the algorithm has at most one active bin of type i. When a bin of type i is opened, it is declared as the active bin of the class, and S_i square "spots", each of which having a size equal to the largest square of class i, are reserved in the bin. Upon the arrival of an item of class i, it is placed in one of the S_i spots of the active bin. If all these spots are occupied, a new bin of type i is opened. This ensures that all bins of type i, except potentially the current active bin, include S_i items.

2.3 Packing Tiny Items

For packing tiny items, the algorithm uses a different approach, proposed by Epstein and van Stee [9]. Briefly, it maintains at most one active bin for placing tiny items. The algorithm maintains a partitioning of the active bin into sub-bins whose sizes are 2^{-i} for non-negative, integer values of i. Upon the arrival of a tiny item of size x, the smallest sub-bin of size $y > x$ is repeatedly partitioned into four sub-bins of size $y/2$, up to the point that further partitioning results in sub-bins of size less than x. At this point, x is placed in one of the resulting partitions. Note that if there is no sub-bin of size $y \geq x$, the active bin is closed, and a new one is opened.

Lemma 1 *[9] Consider the square packing problem (without rotation) in which all items are of size at most $1/M$ for some integer $M \geq 2$. There is an online algorithm (as described above) that creates a packing in which all bins, except possibly one, have an occupied area of size at least $(M^2 - 1)/(M + 1)^2$.*

Class	Side length x	S_i	Occupied Area	Weight	Density
1	$(0.5000, 1.0000]$	1	$> 1(0.250)$=0.250	1	< 4.000
2	$(0.3694, 0.5000]$	4	$> 4(0.136)$=0.544	1/4	< 1.838
3	$(0.3333, 0.3694]$	5	$> 5(0.111)$=0.555	1/5	< 1.801
4	$(0.2697, 0.3333]$	9	$> 9(0.072)$=0.648	1/9	< 1.543
5	$(0.2579, 0.2697]$	10	$> 10(0.066)$=0.660	1/10	< 1.515
6	$(0.2500, 0.2579]$	11	$> 11(0.062)$=0.682	1/11	< 1.466
7	$(0.2139, 0.2500]$	16	$> 16(0.045)$=0.720	1/16	< 1.388
8	$(0.2073, 0.2139]$	17	$> 17(0.042)$=0.714	1/17	< 1.400
9	$(0.2047, 0.2073]$	18	$> 18(0.041)$=0.738	1/18	< 1.355
10	$(0.2000, 0.2047]$	19	$> 19(0.040)$=0.760	1/19	< 1.315
11	$(0.1779, 0.2000]$	25	$> 25(0.031)$=0.775	1/20	< 1.290
12	$(0.1752, 0.1779]$	26	$> 26(0.030)$=0.780	1/26	< 1.282
13	$(0, 0.1752]$		> 0.702	$1.425x^2$	≈ 1.425

Table 2: A summary of item classification, weights, and densities, as used in the definition and analysis of Square-Rotate.

2.4 Analysis

In this section, we prove a competitive ratio of at most 2.306 for Square-Rotate. We use a *weighting function argument*. For each item of size x, we define a weight $w(x) \geq x$ and prove that: (1) the total weight of square items in each bin of the algorithm, except potentially a constant number of them, is at least 1, and (2) the total weight of items in each bin of an optimal packing is at most 2.306. If $w(\sigma)$ denotes the total weight of items in an input sequence σ, then (1) implies that the number of bins opened by the algorithm is at most $w(\sigma) + c$, for some constant value of c, and (2) implies that the number of bins in an optimal packing is at least $w(\sigma)/2.306$. Therefore, the (asymptotic) competitive ratio of the algorithm would be at most 2.306.

Weight assignment. Recall that all bins opened for squares of class i ($1 \leq i \leq 12$), except possibly the last active bin, include S_i squares. We define the weight of items of class i to be $1/S_i$. This way, the total weight of items in bins opened for all squares of classes 1 to 12, except possibly 12 of them (the last bin from each class), is exactly 1. Therefore, (1) holds for bins opened for regular items.

We define the weight of a tiny item of size x as $x^2/0.701 (\approx 1.42x^2)$. All tiny items are of size at most 0.1752. Therefore, by Lemma 1, the occupied area of all bins opened for tiny items (except possibly one of them) will be at least 0.701. This implies their total weight is at least $0.701/0.701 = 1$.

Table 2 gives a summary of the weights of items in different classes. From the above argument, we conclude the following lemma.

Lemma 2 *The total weight of squares in each bin opened by* Square-Rotate*, except possibly 13 of them, is at least 1.*

Next, we provide an upper bound for the total weight of items in a bin of the optimal offline algorithm (Opt). Define the *density* of an item of size x as the ratio between its weight and area, i.e., $w(x)/x^2$. Given the lower bound for the size of each square belonging to class i ($1 \leq i \leq 12$), we can calculate an upper bound for the density of items in each class. For tiny items, the density is simply $1.425x^2/x^2 = 1.425$. Density upper bounds for all classes are reported in Table 2. Defining densities comes in handy in a case analysis used to prove the following lemma.

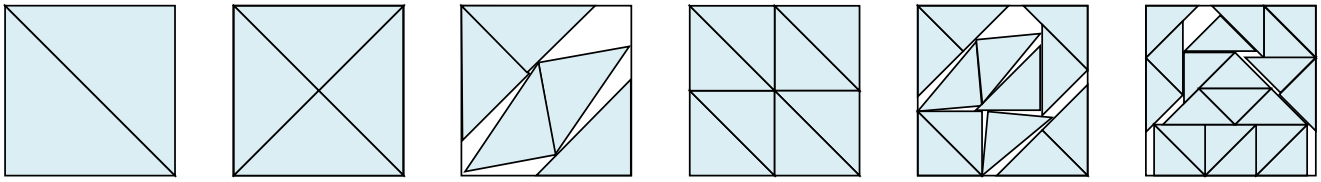
Lemma 3 (Appendix A) *The total weight of items in a bin of* Opt *is less than* 2.306*.*

Proof. (sketch) We consider a bin B of Opt and use case analysis to find an upper bound for the total weight of items in B. The case analysis considers the number of items of classes 1, 2, and 3 in B. In each case, the upper bounds for densities yield an upper bound for the total weight. In the simplest case, when there is no item of class 1 in B, the density of all items will be at most 1.838, and so will be the total weight of all items in B. The presence of an item of class 1 restricts the number and class of other items in B, as captured by 14 sub-cases in the proof of the lemma. Nevertheless, the maximum weight in all cases is at most 2.306, which happens when B contains one item of class 1, three items of class 2, and one item of class 3. □

Provided with the above two lemmas, we can derive the main result of this section.

Theorem 1 *There is an algorithm* Square-Rotate *for the online square packing problem with rotation which achieves a competitive ratio of at most* 2.306*.*

Proof. For an input σ, let $SR(\sigma)$ and $\text{Opt}(\sigma)$ denotes the cost of Square-Rotate and Opt, respectively. Let



(a) Class 1: $x \in (0.7072, 1.0000]$ $t = 1.000$ (b) Class 2: $x \in (0.5593, 0.7072]$ $t = 0.7072$ (c) Class 3: $x \in (0.5000, 0.5593]$ $t = 0.5593$ (d) Class 4: $x \in (0.3828, 0.5000]$ $t = 0.5000$ (e) Class 5: $x \in (0.3056, 0.3828]$ $t = 0.3828$ (f) Class 6: $x \in (0.2843, 0.3056]$ $t = 0.3056$

Figure 3: Placement of regular tan items of class $i \in [1, 6]$ in their respective bin. It is possible to pack i tans of class i into a square bin [11].

$w(\sigma)$ denote the total weight of items of σ. Lemma 2 implies that $SR(\sigma) \leq w(\sigma) + 13$. Meanwhile, Lemma 3 implies that $Opt(\sigma) \geq w(\sigma)/2.306$. From these inequalities, we conclude $SR(\sigma) \leq 2.306\ \textsc{Opt}(\sigma) + c$, where c is a constant independent of the length of σ. □

It is possible to analyze Square-Rotate when all items are of size at most 1/2. In particular, we can establish the following result using the same weighting argument as before and a case analysis slightly different from that of Lemma 3, to get an upper bound for the total weight of items in a bin of optimal packing.

Theorem 2 *[Appendix B] When all items are of size at most 1/2,* Square-Rotate *achieves a competitive ratio of at most* 1.732.

3 Online Tan Packing

This section studies a problem similar to the online square packing problem, called the online tan packing problem, where the sequence of items is formed by isosceles right triangles (half-squares), which we refer to as "tans".

Definition 2 *The input to the* online tan packing with rotation *problem is a sequence* $\sigma = \langle a_1, a_2, \ldots, a_n \rangle$*, where* $a_t \in (0, 1]$ *denote the leg sizes of right isosceles triangles (half-square triangles or tans), that need to be packed into a minimum number of unit bins. The decisions of the algorithm at any time* t *are irrevocable and are made without knowing the values of* $a_{t'}$ *for* $t' > t$*.*

3.1 Half-Square-Rotate Algorithm

We will introduce an online algorithm, called Half-Square-Rotate, that classifies tans by their leg sizes and packs tans of each class separately from other classes. There are seven classes, as presented in Table 3. We refer to items that belong to classes $i \in [1, 6]$ as regular items and those in class 7 as tiny items. Tiny items have sizes in the range $(0, 0.2843]$. For each class $i \in [1, 6]$, the range of items in class i is specified as $(y_i, y_{i-1}]$ (for convenience, we define $y_0 = 1$). The values of y_i are defined so that a certain number T_i of tans of class i can fit in the same bin.

The specific range of item sizes for each class $i \in [1, 6]$ and values of T_i is derived from the best-known or optimal results on the congruent tan packing problem [11], which asks for the minimum size $s(j)$ of a square that can contain j tans of unit leg size. A scaling argument, where the container size is fixed to be 1, gives $t(j)$ values when the goal is to pack j identical tans of maximum leg size $t(j)$ into a unit square. Table 4 provides the scaled best-known/optimal $t(j)$ values for $1 \leq j \leq 20$. These scaled numbers give the specific ranges that we used for classifying items as follows: Tans of class 1 have sizes in the range $(0.7072, 1]$, and we have $y_1 = 0.7072$. Note that exactly $T_1 = 1$ item of class 1 can fit in the same bin (for tans of size ≤ 0.7072, it is possible to pack at least two tans in the bin). For $i \in [2, 6]$, T_i is the number of items of size y_{i-1} that fit in the same bin. For example, for $i = 2$, we have $T_2 = 4$ because $y_1 = 0.7072$, and up to 4 items of size 0.7072 fit in the same bin. Moreover, y_i is defined as the largest value so that $T_i + 1$ items of size y_i cannot fit in the same bin. For example, we have $y_2 = 0.5593$ because, according to Table 4, $T_2 + 1 = 5$ tans of size 0.5593 do not fit in the same bin (with respect to the best known results). The respective range of items for each class, as well as the values of T_i, are presented in Table 3. Figure 3 shows how T_i items of the largest size in class i can fit into a square bin.

3.2 Packing Regular Items

For each class i ($1 \leq i \leq 6$), the algorithm has at most one active bin of type i. When a bin of type i is opened, it is declared as the active bin of the class, and T_i tan "spots", each of which has a size equal to the largest tan of class i, are reserved in the bin. Upon the arrival of an item of class i, it is placed in one of the T_i spots of the active bin. If all these spots are occupied by previous items, a new bin of type i is opened. This ensures that all bins of type i, except potentially the current active bin, include S_i items.

Class	Side length y	T_i	Occupied Area	Weight	Density
1	$(0.7072, 1.0000]$	2	$> 2(0.250)$=0.500	1/2	< 2.000
2	$(0.5593, 0.7072]$	4	$> 4(0.156)$=0.626	1/4	< 1.599
3	$(0.5000, 0.5593]$	5	$> 5(0.125)$=0.625	1/5	< 1.600
4	$(0.3828, 0.5000]$	8	$> 8(0.073)$=0.586	1/8	< 1.706
5	$(0.3056, 0.3828]$	12	$> 12(0.047)$=0.560	1/12	< 1.785
6	$(0.2843, 0.3056]$	20	$> 20(0.040)$=0.808	1/20	< 1.237
Tiny	$(0, 0.2843]$		> 0.557	$1.795(y^2/2)$	1.795

Table 3: A summary of item classification, weights and densities, as used in the definition and analysis of Half-Square-Rotate.

3.3 Packing Tiny Item

To pack tiny items, Half-Square-Rotate uses the same approach as Square-Rotate. Namely, the algorithm maintains partitioning any tiny bin into sub-bins formed by tans of various sizes. The only difference, compared to the algorithm of Epstein and van Stee [9], is that the square bin is divided initially into two tans (of side length 1) instead of four sub-bins. Subsequently, instead of partitioning larger square sub-bins into four sub-squares, we partition large tan sub-bins into *two* smaller tan sub-bins. One crucial observation is that it is possible to partition a tan into two sub-tans, which allows using the same approach as in [9]. Using a similar proof to the one in [9], we can show this adapted algorithm almost entirely packs each bin.

Lemma 4 *[9] Consider the tan packing problem in which all items are of size at most $1/M$ 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 $(M^2-1)/(M+1)^2$.*

3.4 Analysis

We use a weighting argument to prove the competitive ratio of Square-Rotate is at most 1.897. For each tan of size x, we define a weight $w(x)$ as follows. Recall that all bins opened for tans of class i ($1 \leq i \leq 6$), except possibly the last active bin of each class, include T_i tans. We define the weight of items of class i to be $1/T_i$. For a tiny tan of leg size x, we define its weight as $1.795(x^2/2)$ where $x^2/2$ is the area of the tan. Table 3 gives a summary of the weights of the items in different classes. Our definition of weights ensures that all bins opened for regular items, except potentially the last six active bins, have a total weight of 1 since they include i items of weight i. Similarly, our definition of weight for tiny items, paired with Lemma 4, ensures that all tiny bins, except potentially the active one, will have a weight of 1. We can conclude the following lemma.

n	t	opt./b.k.	n	t	opt./b.k.
1	$= 1.0$	opt	11	≈ 0.4143	b.k.
2	$= 1.0$	opt	12	≈ 0.3828	b.k.
3	≈ 0.7072	opt.	13	≈ 0.3720	b.k.
4	≈ 0.7072	opt.	14	≈ 0.3614	b.k.
5	≈ 0.5593	b.k.	15	≈ 0.3536	b.k.
6	≈ 0.5163	b.k.	16	≈ 0.3536	opt.
7	≈ 0.5003	b.k.	17	≈ 0.3367	b.k.
8	$= 0.5$	opt.	18	≈ 0.3333	opt.
9	≈ 0.4531	b.k.	19	≈ 0.3124	b.k.
10	≈ 0.4179	b.k.	20	≈ 0.3056	b.k.

Table 4: Optimal (opt.) or best-known (b.k.) $t(j)$ values for $1 \leq j \leq 20$ when the goal is to pack j identical tans of the largest leg size $t(j)$ into a unit square. Values of $t(j)$ are the scaled values of the known results on congruent tan packing [11].

Lemma 5 *[Appendix C] The total weight of tans in each bin opened by* Half-Square-Rotate*, except possibly 7 of them, is at least 1.*

Using a case analysis, which is similar to that of Lemma 3 and is based on investigating the density of items in a bin of Opt, we can find an upper bound for the total weight of items in any bin of the optimal offline algorithm Opt.

Lemma 6 (Appendix D) *The total weight of items in a bin of* Opt *is less than* 1.897*.*

Theorem 3 *The* Half-Square-Rotate *algorithm for the online tan packing with rotation achieves a competitive ratio of at most* 1.897*.*

Proof. For an input σ, let $HR(\sigma)$ and $\textsc{Opt}(\sigma)$ denote the cost of Half-Square-Rotate and Opt, respectively. Let $w(\sigma)$ denote the total weight of items of σ. Lemma 5 implies that $HR(\sigma) \leq w(\sigma) + 7$. Meanwhile, Lemma 6 implies that $\textsc{Opt}(\sigma) \geq w(\sigma)/1.897$. From these inequalities, we conclude $HR(\sigma) \leq 1.897\ \textsc{Opt}(\sigma) + c$, for some constant $c \in O(1)$ which proves an upper bound 2.306 for the asymptotic competitive ratio of Square-Rotate. □

Acknowledgements

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) [funding reference number DGECR-2018-00059].

References

[1] S. Angelopoulos, S. Kamali, and K. Shadkami. Online bin packing with predictions. In *Proceedings of the 31st Internation Joint Conference on Artificial Intelligence (IJCAI)*, 2022.

[2] J. Balogh, J. Békési, G. Dósa, L. Epstein, and A. Levin. A new and improved algorithm for online bin packing. In *Proceedings of the 26th Annual European Symposium on Algorithms (ESA)*, volume 112, pages 5:1–5:14, 2018.

[3] J. Balogh, J. Békési, G. Dósa, L. Epstein, and A. Levin. A new lower bound for classic online bin packing. *Algorithmica*, 83(7):2047–2062, 2021.

[4] 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.

[5] H. I. Christensen, A. Khan, S. Pokutta, and P. Tetali. Approximation and online algorithms for multidimensional bin packing: A survey. *Computer Science Review*, 24:63–79, 2017.

[6] E. G. Coffman, M. R. Garey, and D. S. Johnson. Approximation algorithms for bin packing: A survey. In *Approximation algorithms for NP-hard Problems*. 1997.

[7] E. G. Coffman Jr., J. Csirik, G. Galambos, S. Martello, and D. Vigo. Bin packing approximation algorithms: survey and classification. In *Handbook of Combinatorial Optimization*, pages 455–531. 2013.

[8] L. Epstein. Two-dimensional online bin packing with rotation. *Theor. Comput. Sci.*, 411(31-33):2899–2911, 2010.

[9] L. Epstein and R. van Stee. Optimal online bounded space multidimensional packing. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 214–223, 2004.

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

[11] E. Friedman. Packing unit squares in squares: A survey and new results. *The Electronic Journal of Combinatorics*, pages 1–24, 2000.

[12] G. Galambos and G. J. Woeginger. On-line bin packing - A restricted survey. *Math. Methods Oper. Res.*, 42(1):25–45, 1995.

[13] M. R. Garey and D. S. Johnson. Approximation algorithms for bin packing problems - a survey. In G. Ausiello and M. Lucertini, editors, *Analysis and Design of Algorithms in Combinatorial Optimization*, pages 147–172. Springer, 1981.

[14] X. Han, D. Ye, and Y. Zhou. A note on online hypercube packing. *Central European Journal of Operations Research*, 18(2):221–239, 2010.

[15] S. Heydrich and R. van Stee. Beating the harmonic lower bound for online bin packing. In *Proceedings of the 43rd International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 41:1–41:14, 2016.

[16] R. Hoberg and T. Rothvoss. A logarithmic additive integrality gap for bin packing. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2616–2625, 2017.

[17] S. Kamali and A. López-Ortiz. Almost online square packing. In *Proceedings of the 26th Canadian Conference on Computational Geometry (CCCG)*, 2014.

[18] S. Kamali and A. López-Ortiz. All-around near-optimal solutions for the online bin packing problem. In *Proceedings of the 26th International Symposium on Algorithms and Computation (ISAAC)*, volume 9472, pages 727–739, 2015.

[19] S. Kamali, A. López-Ortiz, and Z. Rahmati. Online packing of equilateral triangles. In *Proceedings of the 27th Canadian Conference on Computational Geometry (CCCG)*, 2015.

[20] S. Kamali and P. Nikbakht. Cutting stock with rotation: Packing square items into square bins. In *Proceedings of the 14th International Conference on Combinatorial Optimization and Applications (COCOA)*, pages 530–544, 2020.

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

[22] S. S. Seiden and R. van Stee. New bounds for multidimensional packing. *Algorithmica*, 36(3):261–293, 2003.

	C1	C2	C3	Sum of Weights (W)	Sum of Areas (A)	Remaining Area ($A_r = 1-A$)	Weight of Items in the Remaining Area ($W_r = A_r \times 1.543$)	Total Weight of Items in the Bin ($W_{max} = W + W_r$)
Number of items of each class in an OPT bin	1	0	0	1.00	> 0.250	< 0.750	< 1.157	< 2.157
	1	0	1	1.20	> 0.361	< 0.639	< 0.986	< 2.186
	1	0	2	1.40	> 0.472	< 0.528	< 0.815	< 2.215
	1	0	3	1.60	> 0.583	< 0.417	< 0.644	< 2.244
	1	0	4	1.80	> 0.694	< 0.306	< 0.472	< 2.272
	1	1	0	1.25	> 0.386	< 0.614	< 0.948	< 2.198
	1	1	1	1.45	> 0.497	< 0.503	< 0.776	< 2.226
	1	1	2	1.65	> 0.608	< 0.392	< 0.605	< 2.255
	1	1	3	1.85	> 0.719	< 0.281	< 0.434	< 2.284
	1	2	0	1.50	> 0.522	< 0.478	< 0.738	< 2.238
	1	2	1	1.70	> 0.633	< 0.367	< 0.566	< 2.266
	1	2	2	1.90	> 0.744	< 0.256	< 0.395	< 2.295
	1	3	0	1.75	> 0.658	< 0.342	< 0.528	< 2.278
	1	3	1	1.95	> 0.769	< 0.231	< 0.356	< 2.306

Table 5: The fourteen possible cases for a combination of items of class 2 (C2) and 3 (C3) together with an item x of class 1 (C1) in a bin B. Here, *sum of weights* (W) and *sum of areas* (A) indicate, respectively, the total weight and area of items of the first three classes in B. *Remaining area* is the area left in the bin that is used for packing items of class 4 or higher. *Weight of items in the remaining area* is an upper bound for the total weight of items of class 4 or higher in B (these items have a density of no more than 1.543). Finally, *total weight of items in the bin* indicates the sum of weights of all items (from all classes) in B.

Appendix

A Proof of Lemma 3

Proof. We use the following case analysis:

Case 1: Assume there is no item of class 1 in B. Since the density of items of other classes are less than 1.838, even if B is fully packed with items of the largest density, the total weight of items cannot be more than 1.838, which is less than 2.306.

Case 2: Assume there is one item x of class 1 (note that no two items of class 1 fit in the bin). Without loss of generality, we assume the size of x is $1/2+\epsilon$, where ϵ is a small positive value greater than zero. Clearly, a larger size for x does not increase the total weight of other items in B because it would leave less space to occupy more items in the bin (while the weight of x stays 1). Next, we consider all possible cases in which we have some items of class 2 and 3 together with x in B. As presented in Table 5, there will be 14 sub-cases to analyze. To see how we reach these 14 sub-cases, first note that it is not possible to accommodate four or more items of class 2 in addition to x in B (i.e., a total number of 5 or more items from these classes 1 and 2). This is because no five items with size larger than 0.3694 can fit in B [11]. A similar argument shows that we cannot have six or more items from classes 1, 2, and 3 together in a bin; otherwise, we could accommodate six identical squares of size strictly larger than 0.3333, which is a contradiction to the fact that no six items of size larger than 0.3333 can fit in the same bin [11]. In summary, the 14 sub-cases summarized in Table 5 cover all possibilities for items of the first three classes in Case 2.

According to Table 2, the density of items belonging to class i ($4 \leq i \leq 12$) as well as tiny items is at most 1.543 (which is the density of class-4 items). Using a similar argument made for Case 1, we suppose that, after placing a certain number of items of class 2 and 3 beside x in B, in each sub-case, we can fill the remaining space of B with the items of the maximum density 1.543. This allows us to calculate an upper bound for the maximum total weight of items in B for each of the sub-cases. The resulting bounds for each sub-case can be found in the last column of Table 5, where the maximum upper bound among all sub-cases is 2.306, which happens when we have one item of class 1 in B together with three items of class 2 and one item of class 3.

As a result, in both Case 1 and Case 2, the total weight of items in B cannot be more than 2.306. □

B Proof of Theorem 2

Proof. We employ the same approach and weighting function as the one we used to analyze the upper bound of Square-Rotate for the general setting, except that we exclude items of class 1, that is, items of size more than 1/2. By Lemma 2, the total weight of items in each bin of Square-Rotate, except for a possibly a constant number of them, is at least 1. Therefore, to prove the theorem, it suffices to show the total weight of items in any bin B of an optimal packing is at most 1.732. To study the maximum weight of items in B, we consider all possible combinations of items from classes 2 and 3 in B. For that, we consider the following two limitations: (i) we cannot have more than four items of class 2 in B; otherwise, we could accommodate five squares of size more than 0.3694, which is not possible [11]. (ii) we cannot have more than six items of class 2 or 3 in B. Otherwise, one could place six squares of size more than 0.3333, which is known to be impossible [11]. Altogether, we will have 19 cases to consider as presented in Table 6.

	C2	C3	Total Weight of Items in the Bin	C2	C3	Total Weight of Items in the Bin
Number of items of each class $c \in \{2,3\}$ in B.	0	0	< 1.543	1	4	< 1.698
	0	1	< 1.572	2	0	< 1.623
	0	2	< 1.601	2	1	< 1.652
	0	3	< 1.629	2	2	< 1.681
	0	4	< 1.658	3	0	< 1.664
	0	5	< 1.687	3	1	< 1.692
	1	0	< 1.583	3	2	< 1.721
	1	1	< 1.612	4	0	< 1.704
	1	2	< 1.641	4	1	< 1.732
	1	3	< 1.669			

Table 6: Maximum total weight of items of size $\in (0, 1/2)$ in a bin B of an optimal packing in all nineteen possible cases in which there is no item of class 1 but a combination of items of class 2 (C2) and 3 (C3) in B.

We have used the same method as in Lemma 3 to calculate the upper bound for the total weight of items in each case. Table 6 summarizes the final results for all cases. The maximum weight, 1.732, happens when we have four items of class 2 together with one item of class 3 packed in B. □

C Proof of Lemma 5

Proof. In every bin of class i ($1 \leq i \leq 6$), except possibly the last bin, there are T_i items, each having a weight of $1/T_i$. As a result, the total weight of the items in each bin of such classes, except possibly 6 of them, is exactly $T_i \times 1/T_i = 1$. For bins of tiny items, we know, by Lemma 4 and considering $1/M = 0.2843$, that the occupied area of all bins, except possibly the last one, will be at least 0.557. Therefore, the total weight of items in such a bin is at least $W = 1.795 \times 0.557 = 1$. □

D Proof of Lemma 6

Proof. We first define the *density* of a tan item of leg size x as the ratio between its weight and its area, i.e., $w(x)/(x^2/2)$. Given the lower bound for the leg size of items in each class i ($1 \leq i \leq 6$), and, hence, their area, we can calculate an upper bound for the density of each item in the class. For tiny items, the density is simply equal to $1.795(x^2/2)/(x^2/2) = 1.795$. The densities of items from different classes are reported in Table 3. In what follows, we use a case analysis approach to prove that the total weight of items in a bin B of an optimal packing is at most 1.897. There are three cases to consider: either 0, 1, or 2 tans of class 1 exist in the bin. Note that it is not possible to accommodate 3 or more items of class 1 in a bin because their total area would exceed 1.

Case 1: No class-1 item in B: Since the density of items of class 2 or larger is at most 1.795, even if all area of B is filled with items of the largest density, the total weight of items cannot exceed 1×1.795 which is less than 1.897.

Case 2: Exactly one class-1 item in B: If there is exactly one item of class 1 (with weight 1/2 and the area of at least $(0.7072 \times 0.7072)/2 = 0.25$) in B, the remaining area in the bin will be at most 0.75. Items of classes other than class 1 have a density of at most 1.795. Even if we fill the remaining area with such items of the highest density, the total weight of items in B will not exceed 1.346. As a result, the total weight of items in B cannot be more than $1/2 + 1.346 = 1.846$, which is less than 1.897.

Case 3: Exactly two class-1 items in B: If there exists exactly two items of class 1 with weight 1/2 (and the total weight of 1) and total area of at least $2 \times (0.7072 \times 0.7072)/2 = 0.50$, the remaining area in B will be at most 0.50. If this remaining area is filled by items of the highest density, which are items of the last class with a density of at most 1.795, the total weight of items in B will not be more than $1 + 0.50 \times 1.795 = 1.897$.

In conclusion, the total weight of items in a bin B of an optimal packing cannot be more than 1.897 in all cases. □