Online Square Packing with Predictions*

Stephane Durocher† Shahin Kamali‡ Pouria Zamani Nezhad§

Abstract

Square packing is a geometric variant of the classic bin packing problem, which asks for the placement of squares of various lengths into a minimum number of unit squares. In this work, we study the online variant of the problem in which the input squares appear sequentially, and each square must be packed before the next square is revealed. We study the problem under the *prediction setting*, where the online algorithm is enhanced with a potentially erroneous prediction about the input sequence. We design an online algorithm that receives predictions concerning the sizes of input squares and analyze its *consistency* (the competitive ratio assuming no error in the predictions) and *robustness* (the competitive ratio under adversarial error). In particular, our algorithm has consistency $1.77\bar{9}$ and robustness at most 5.89. These results show improvements over the best previous algorithm [24], designed for perfect predictions, with a consistency of 1.84 and a robustness of at least 21.

1 Introduction

Given a multiset of n square items, each with a fixed sidelength in $(0, 1]$, the square packing problem seeks to assign each item to a unit square bin, such that the number of bins is minimized. We consider orthogonal packings, in which each item's interior is contained in the interior of its assigned bin, each item's edges are oriented parallel to its assigned bin's edges (axis parallel), and no two items' interiors in the same bin intersect (pairwise interior-disjoint). We refer to a square's sidelength as its *size*. This is a geometric variant of the classic bin packing problem. Similarly to the bin packing problem, square packing is NP-hard, but admits an Asymptotically Polynomial-Time Approximation Scheme (APTAS) [10].

We consider *online square packing*, in which input square items are revealed one at a time in an online sequence. Upon receiving each item, an algorithm must assign it to a bin with sufficient space immediately, without any knowledge about future items. Bin assignments are irrevocable. The standard measure for evaluating an online algorithm is the *asymptotic competitive ratio*, which compares the cost of the online algorithm against the optimal (offline) cost in the worst case. For the online square packing problem, the asymptotic competitive ratio of an online algorithm Alg is

$$\lim_{n\to\infty} \sup_{\sigma:|\sigma|=n} \frac{|\textsc{Alg}(\sigma)|}{|\textsc{Opt}(\sigma)|},$$

where $|\textsc{Alg}(\sigma)|$ denotes the number of bins used by Alg to pack the input sequence σ, and $|\textsc{Opt}(\sigma)|$ denotes the minimum number of bins required by any (optimal) packing of σ. We refer to asymptotic competitive ratio simply as *competitive ratio*. No online square packing algorithm can achieve a competitive ratio better than 1.75 [8], while the best previous algorithm has a competitive ratio of at most 2.0885 [20].

Square packing has been studied under the *advice setting*, which relaxes the assumption that the algorithm has no advance information about the input sequence, and provides the online algorithm access to error-free information about the input sequence called *advice* before packing the first item in the input sequence [18]. The objective is to quantify trade-offs between the competitive ratio and the number of bits of advice. For square packing, there is an online algorithm that achieves a competitive ratio of at most 1.84 with $O(\log n)$ bits of advice [24]. Unfortunately, this result has little practical significance, partially because the advice is assumed to be error-free.

In this paper, we study the online square packing problem under a recently developed and more practical model, which seeks to leverage *predictions* about the input sequence [28]. Specifically, the algorithm can access some machine-learned information about the input sequence. Unlike with advice, predictions may be erroneous. Moreover, the predictions should be *efficiently learnable* (e.g., via sampling the input sequence). The objective is to design an algorithm that performs well if the prediction is accurate while maintaining a good competitive ratio even when the prediction is highly erroneous (i.e., adversarial). We refer to the competitive ratio of an online algorithm with an error-free prediction as its *consistency* and to the competitive ratio with an adversarial prediction as its *robustness* [28]. Several online optimization problems have been stud-

*This work is funded in part by the Natural Sciences and Engineering Research Council of Canada (NSERC).

†University of Manitoba, stephane.durocher@umanitoba.ca
‡York University, kamalis@yorku.ca
§University of Manitoba, zamaninp@myumanitoba.ca

ied under the prediction model, including bin packing [3, 4], scheduling [2, 12, 30, 32], knapsack [11, 23, 33], caching [28, 31], matching [5, 25, 26], and various graph problems [7, 9, 14, 15, 19]. See also the survey by Mitzenmacher and Vassilvitskii [29] and the collection at [1].

1.1 Contribution

We study the square packing problem under a setting where the online algorithm exploits natural predictions concerning the *frequency* of item sizes. We classify square items based on their sizes and consider predictions on the number of items within certain classes. To be more precise, predictions specify the number of items in the input sequence with side-lengths in the ranges $(2/3, 4/5]$, $(3/5, 2/3]$, $(11/20, 3/5]$, $(1/2, 11/20]$, and $(1/3, 2/5]$. For an input sequence of n items, these predictions can be encoded in $O(\log n)$ bits, and they are Probably Approximately Correct (PAC)-learnable [13]. We design an algorithm, named Reserve-and-Pack (RAP), which makes use of the above predictions. Our results can be summarized as follows:

- We show that RAP has a consistency of $1.77\bar{9}$ (Theorem 6). In other words, when predictions are error-free (they are advice), the competitive ratio of RAP is at most $1.77\bar{9}$. This result is an improvement over the algorithm of [24], Almost-Online-Square-Packing (AOSP), which has a consistency of 1.84. Both algorithms use a prediction (advice) of size $O(\log n)$.
- We show that the robustness of AOSP is at least 21 (Theorem 7). Moreover, we prove that the robustness of RAP is at most $100/17 \lessapprox 5.89$ (Theorem 8). In other words, RAP dominates AOSP regarding both consistency and robustness. This is due to its improved item classification and increased flexibility in adapting to patterns in the input rather than overly relying on the predicted patterns.

2 Reserve and Pack (RAP) Algorithm

In this section, we present our algorithm Reserve-And-Pack (RAP). RAP works by classifying items based on their sizes and receiving predictions about the frequency (number) of items from certain classes with larger sizes. The algorithm proceeds by reserving a *placeholder* for each of these items in anticipation of their arrival. We point out that AOSP receives similar predictions and also uses placeholders [24]. RAP improves over AOSP by refining the item classification, which results in improved consistency. In addition, RAP only reserves space for certain items of larger size, unlike AOSP, which forms an offline packing of the predicted input and reserves placeholders for *all* items (except for "tiny" items). In other words, RAP's reliance on prediction is minimal compared to AOSP. As a result, it has superior robustness in the case of erroneous predictions.

Class	Interval	Class	Interval
$1a$	$(4/5, 1]$	$2a$	$(2/5, 1/2]$
$1b$	$(2/3, 4/5]$	$2b$	$(1/3, 2/5]$
$1c$	$(3/5, 2/3]$	3–29	$(1/i+1, 1/i]$
$1d$	$(11/20, 3/5]$	30	$(0, 1/30]$
$1e$	$(1/2, 11/20]$		

Table 1: Item classification used by RAP

Item Classification. RAP classifies items into 30 classes based on their sizes. For $i \in [1..29]$, items with size in the range $(\frac{1}{i+1}, \frac{1}{i}]$ belong to class i. Items with sizes in the $(0, 1/30]$ form the 30th class and are called *tiny* items. Items of Class 1, which are larger than 0.5, are called *large* items and are further divided into 5 subclasses $1a, 1b, 1c, 1d$, and $1e$ with sizes corresponding to the intervals $\left(\frac{4}{5}, 1\right]$, $\left(\frac{2}{3}, \frac{4}{5}\right]$, $\left(\frac{3}{5}, \frac{2}{3}\right]$, $\left(\frac{11}{20}, \frac{3}{5}\right]$, and $\left(\frac{1}{2}, \frac{11}{20}\right]$, respectively. Similarly, items of Class 2 are called *medium* items and are further divided into two subclasses $2a$ and $2b$ with respective associated intervals $\left(\frac{2}{5}, \frac{1}{2}\right]$ and $\left(\frac{1}{3}, \frac{2}{5}\right]$. Table 1 summarizes defined classes and their corresponding size intervals.

In addition to items, each bin of RAP has a type, which is determined by the class of items it contains. Specifically, *LM-bins* contain a large or a medium item, say of type ℓ, and smaller items of the same type $t \geq 3$, in which case the bin is referred to as a $\langle \ell, t \rangle$ bin. For example, when $\ell = 2b$ and $t = 10$, the LM-bin is of type $\langle 2b, 10 \rangle$ and only contains medium items of type $2b$ and small items of type 10 (see Figure 1c). When the large item is of type $\ell = 1e$ and the small item is of type $t = 4$, an LM-bin is called a *critical bin* and is allowed to contain items of a third type t', in which case it is referred to as a $\langle 1e, 4, t' \rangle$ bin. In addition to LM-bins, RAP maintains *harmonic* bins that only include items of the same type, say t, in which case the bin is said to be a harmonic-t bin. A harmonic-t bin is said to be a *large harmonic bin* if t is a large or medium type and *small harmonic bin* otherwise. We note that large harmonic bins may change their type to become LM-bins (when a small or tiny item is placed in them).

Preprocessing. RAP relies on predictions about the number of items belonging to large and medium types. Specificly, RAP uses a frequency vector $\mathbf{f} = \langle f_{1b}, f_{1c}, f_{1d}, f_{1e}, f_{2b} \rangle$, where f_t is the predicted number of class t items in the input sequence σ. Note that the predictions do not concern $1a$ and $2a$ items as they are "easy to pack"; i. e., they can be packed into almost

full bins without involving other items, as will be clarified later. Before packing the input sequence, for each predicted frequency f_t, RAP creates f_t *placeholders* of class t, that is, a reserved space of equal size to the maximum size of class t items. RAP assigns placeholders of each class in separate bins, while groups of four $2b$ placeholders share one bin. These bins are "virtually" open, and they contribute to the cost of RAP only after an item is placed into them. We assume placeholders are positioned on the top-left of their respective bins.

Online Packing. When possible, RAP places large and medium items in placeholders reserved in the preprocessing step. This is done through a procedure called ASSIGNLM that we will describe shortly. Small and tiny items, however, are packed into designated *containers* that are formed and placed in an online manner. A container is a dedicated space that can accommodate either a single small item or multiple tiny items. Upon the arrival of a small or tiny item of class c, it is placed in an available container of the same class using the corresponding ASSIGNSMALL or ASSIGNTINY procedures, which will also be described later. If no such container exists, RAP creates a new set of class c containers using a subroutine called RESERVE.

The RESERVE subroutine, for any given class $c \geq 3$ (small or tiny), first attempts to place containers of class c in a critical bin, and if not possible, a large harmonic bins using the *L-shape tiling* of [24]. This involves placing containers of type c in the non-reserved space of the bin in a greedy manner in columns and rows that collectively form an "L"-shape. If no critical or large harmonic bin is available, it opens a new small harmonic bin of type c. More precisely, RESERVE takes the following steps to create new containers of type c:

1. If $c \geq 5$ and there is a critical bin B (i.e., a bin of type $\langle 1e, 4\rangle$), add containers of type c to B, using L-shape tiling, and update the type of B to $\langle 1e, 4, c\rangle$.
2. If a large harmonic bin B of type ℓ with a reserved space of r is available, and $1/c \leq 1-r$, use L-shape tiling to place containers of class c to B, and update the type of B to $\langle \ell, c\rangle$. Here, "available" means that B does not contain any other containers.
3. Otherwise, open a new harmonic bin of type c and place c^2 containers of class c into it.

RAP consists of three main components: ASSIGNLM, ASSIGNSMALL, and ASSIGNTINY, packing corresponding items of large/medium, small, and tiny classes.

- ASSIGNLM assigns each $1a$, or four $2a$ items into a single bin. In addition, it packs an item of class $c \in \{1b, 1c, 1d, 1e, 2b\}$ into any available placeholder of class c. If no placeholder is available (due to a prediction error), it opens a new bin and declares it as a large harmonic bin of type c.
- ASSIGNSMALL packs small items of class c into the next empty container of size $1/c$. If no such container is available, it creates a new set of containers by invoking the RESERVE subroutine.
- ASSIGNTINY places tiny items into *tiny containers*. A tiny container is of size $1/5$ and is dedicated to tiny items. As before, if no tiny container exists, ASSIGNTINY first creates a new set of tiny containers by calling RESERVE. We borrow the algorithm of [22] to pack tiny items into tiny containers. The algorithm repeatedly splits tiny containers into smaller sub-containers to pack a tiny item of size s into a sub-container of size $1/2^k$, where k is the largest integer such that $s \leq 1/2^k$.

Figure 1 shows examples of bins packed by RAP. In particular, Figures 1a and 1e are bins that used to be critical and had their types changed after receiving additional small containers.

3 Consistency Analysis

Overview. In this section, we analyze the consistency of RAP. First, we use the following lemma to show that all tiny containers, except possibly the last one, are almost full.

Lemma 1 *[22] Consider the square packing problem where 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 a constant number of them, have an occupied area of size at least $(M^2-1)/(M+1)^2$.*

In our context, tiny items of size at most $1/30$ are packed into containers of size $1/5$. With a scaling argument, the above result applies with $M = 6$. Another ingredient in our proof is a lower bound for the number of containers of a given class placed into a bin using L-shape tiling. In particular, we will use the following lemma:

Lemma 2 *Consider a square space S of sidelength $s \in \{0.75, 1\}$, from which a square space of size $r < s$ is reserved. It is possible to pack $2ki - k^2$ containers of size c in the remainng area of S, where $i = \lfloor s/c \rfloor$ and $k = \lfloor (s-r)/c \rfloor$.*

Proof. Consider an empty bin of size s; it fits i^2 containers of size c, where $i = \lfloor s/c \rfloor$. However, $(i-k)^2$ of these containers overlap with the reserved space, where

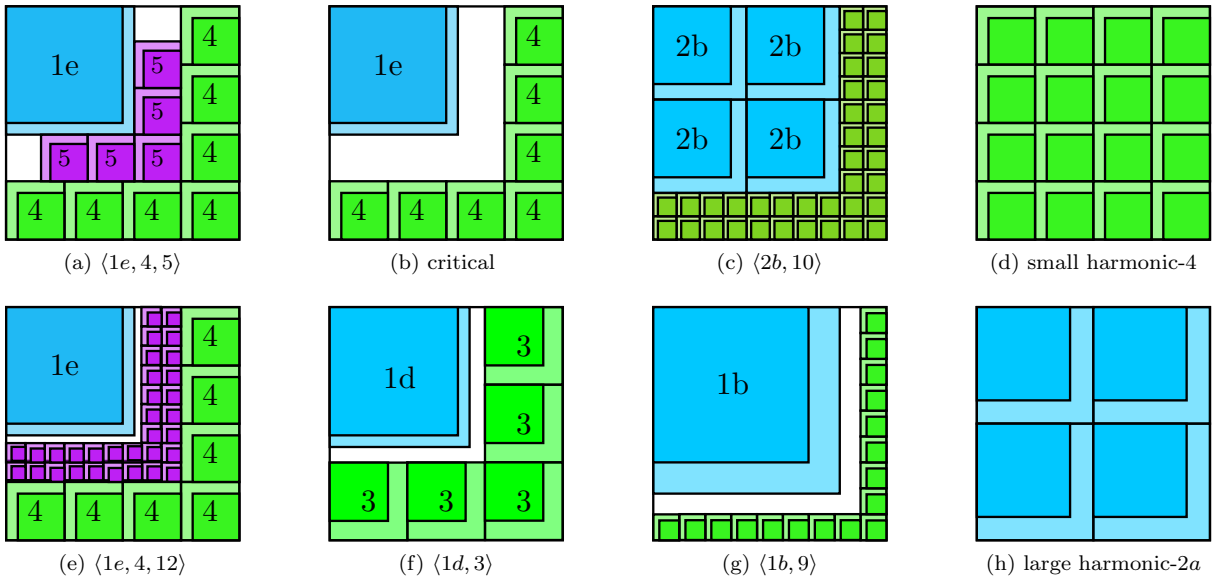

Figure 1: Examples of possible bin types of RAP. The light-colored areas represent reserved spaces, while dark-colored areas show the minimum occupied area by items of each specified type. Blue areas are placeholders, while green and purple areas are containers. Purple containers are extra containers added to critical bins.

$k = \lfloor (s-r)/c \rfloor$. It implies that there is room to add

$$\begin{aligned} &\lfloor s/c \rfloor^2 - (\lfloor s/c \rfloor - \lfloor (s-r)/c \rfloor)^2 \\ &\quad = i^2 - (i-k)^2 \\ &\quad = k(2i-k) \\ &\quad = 2ki - k^2 \end{aligned}$$

containers to this bin which completes the proof. □

When using L-shape tiling to place containers in large harmonic bins (packing green containers in Figure 1), we have $s = 1$. When using L-shape tiling to place small containers in critical bins (packing purple containers in the figure), we have $s = 0.75$. Table 2 in Appendix presents the minimum occupied area of all LM bins by applying Lemmas 1 and 2.

To prove an upper bound of $1.77\bar{9}$ for the consistency of RAP, we consider three possibilities for the final packings of RAP. The first case is when all bins in the final packing contain a large or medium item. In other words, no small harmonic bin is opened. In this case, we use a standard *weighting technique* [6] to prove the upper bound for the competitive ratio (Lemma 3). The second case is when the final packing has a harmonic bin of class $c \geq 5$. In this case, we show that all bins are sufficiently full to prove the upper bound (Lemma 4). Finally, the third case concerns situations in which there is no harmonic bin of class $c \geq 5$ in the final packing. In this case, we use a more complicated weighting argument that involves solving an integer program to prove the upper bound (Lemma 5).

Case I: No small harmonic bin. Suppose no bin is opened for containers of tiny or small items. We assign a weight of $w(x)$ to an item of size x, and prove that (i) the total weight of items in any bin of RAP, except possibly a constant number of them, is at least 1, while (ii) the weight of items in any bin of OPT is at most α. Therefore, if W denotes the total weight of all items in the input, we can write $\text{RAP}(\sigma) \leq W + c$, for some constant c, while $\text{OPT}(\sigma) \geq \lceil W/\alpha \rceil$, which yields to a competitive ratio of at most α.

Lemma 3 *Suppose there is no small harmonic bin of class $c \geq 3$, in the final packing of* RAP. *Then, the competitive ratio of* RAP *is at most* 1.75.

Proof. We assign to all items of class $c \geq 3$ a weight of 0. Large items have a weight of 1, and medium items have a weight of 1/4. All bins in the final packing of RAP, except possibly two of them, either contain a large item or four medium items. Therefore, all bins opened by RAP have a total weight of at least 1. It follows that $|\text{RAP}(\sigma)| \leq W$, where W is the total weight of all items in σ. On the other hand, a bin of $\text{OPT}(\sigma)$ may contain three medium and one large item, e. g., an item of size $0.5 + \epsilon$ and three items of size $0.5 - \epsilon$. Moreover, no more than one large item and four medium items fit into the same bin, giving a maximum total

weight of 1.75 for items in any bin of OPT. Therefore, $|\text{OPT}(\sigma)| \geq W/1.75$, resulting in a competitive ratio of at most 1.75 for RAP. □

Case II: There is a small harmonic bin of type $t \geq 5$. Suppose there is at least one small harmonic bin of type $t \geq 5$ in the final packing. We show that all opened bins, except possibly a constant number of them, have an occupied area of at least 9/16, which indeed guarantees a competitive ratio of at most 16/9.

Lemma 4 *Assume there is a harmonic-t bin in the final packing of* RAP*, for some $t \geq 5$. Then, the occupied area in all bins, except possibly a constant number of them, is at least* 9/16.

Proof. We begin by observing that large harmonic bins with an item of Class $1a$ or four items of Class $2a$ have an occupied area of at least $16/25 = 0.64$. Next, we show that the final packing of RAP contains no large harmonic or critical bins. For the sake of contradiction, suppose there is a large harmonic or critical bin B of type ℓ, and note that B receives a placeholder in the preprocessing step. Therefore, before opening small harmonic bins, specifically t-harmonic bins, RESERVE must use L-shape tiling to place containers of class t in B. This would change the type of B to $\langle \ell, t \rangle$, contradicting its final type being a large harmonic bin of type ℓ. We conclude that the final packing of RAP only contains LM bins and small harmonic bins.

Note that RESERVE procedure adds new containers to one bin upon each call, and it is only invoked once there are no empty containers left for an arrived item. It implies there is at most a constant number of empty containers at any time during the execution of the algorithm. Therefore, harmonic bins of type $t' \geq 3$, except possibly a constant number of them (which have empty containers), each contains t'^2 items of class t' and a minimum occupied area of at least $t'^2/(t'+1)^2 \geq 9/16$. Similarly, by Lemma 1, each tiny bin has a minimum occupied area of $35/49 > 9/16$. Finally, Lemmas 1 and 2 show that all LM-bins, except those with empty containers, each has a minimum occupied area of at least 9/16, as reported in Table 2 [Appendix]. Therefore, all bins in the final packing of RAP, except possibly a constant number of them, have a minimum occupied area of at least 9/16. □

Case III: All small harmonic bins are of Class 3 or 4. We consider the case where there is no small harmonic bin of type $t \geq 5$ in the final packing of RAP, while there is a small harmonic of Type 3 or 4. In this case, there are critical bins in the packing, and not enough small items of class $t \geq 5$ were revealed to cover the "wasted" space in these critical bins. Similar to Case I, we use a weighting argument to prove the following lemma.

Lemma 5 *Assume there is no small harmonic bin of type $t \geq 5$ in the final packing, while there is a small harmonic-t′ bin, where $t' < 5$. Then, the competitive ratio of* RAP *is at most* $1.77\bar{9}$.

Proof. We assign a fixed weight to all items of the same class except for tiny items, which receive a weight proportional to their sizes. We define $\mathbf{w} = \langle w_1, \ldots, w_{34} \rangle$, where w_i is the weight assigned to items of (sub)class $i \leq 34$. Additionally, we assign a weight of $w(x) = d \times s(x)^2$ to a tiny item x, where $s(x)$ is the size of x and d is a fixed constant called the *density* of tiny items. The specific weights are specified in Table 3 [Appendix]. These weights are defined in a way to guarantee a total weight of at least 1 for all bins in the final packing, except possibly a constant number of them. We use an integer program to prove an upper bound on the weight of bins in OPT. Let $\mathbf{t} = \langle t_1, \ldots, t_{35} \rangle$ denote the maximum size of items in each class, in decreasing order; that is, for $i \leq 7$, t_i denotes the maximum size of large or medium items of subclass i and, for $j \geq 7$, t_j denote the maximum size of items of class $j-5$. Let x_i denote the number of items of class i in a bin, say B, packed by OPT. To maximize the weight of the items in B, we can write the following integer program:

maximize:

$$\alpha = \sum_{i=1}^{34} w_i \cdot x_i + d \cdot \left(1 - \sum_{i=1}^{34} x_i \cdot t_{i+1}^2\right)$$

subject to:

$$\sum_{i=1}^{34} x_i \cdot t_{i+1}^2 \leq 1 \tag{1}$$

$$\sum_{i=1}^{34} \lfloor t_{i+1} \cdot (u+1) \rfloor^2 \cdot x_i \leq u^2, \quad \forall u \in \{1, \ldots, 60\} \tag{2}$$

$$x_i \geq 0 \text{ and } x_i \in \mathbb{Z} \quad , \quad \forall i \in \{1, \ldots, 34\}$$

The first component of the objective function is the total weight of all non-tiny items, and the second is an upper bound on the weight of all tiny items in B. Constraint 1 ensures the total area of non-tiny items of B does not exceed 1, and Constraint 2 ensures all items fit into B without overlap. This constraint must hold because, for any integer $u \geq 1$, a bin cannot contain more than u^2 squares of size above $\frac{1}{u+1}$. Figure 2 represents the optimal solution to the integer program resulting in a $1.77\bar{9}$ upper bound on α which completes the proof. □

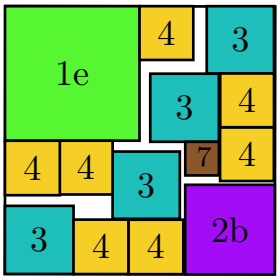

Figure 2: The packing that maximizes the total weight of items in a bin of Opt, as discussed in Lemma 5

Wrapping up. Our results imply the following upper bound for the consistency of Rap.

Theorem 6 Rap *has a consistency of at most* $1.77\bar{9}$.

Proof. Lemmas 3 and 4 show that the competitive ratio is at most $1.77\bar{9}$ in Cases I and III. In case II, Lemma 5 shows that Rap opens at most $16W/9 + c$ bins, where W is the total area of all items, and c is a constant. Given that Opt opens at least W bins, the competitive ratio, in this case, is at most $16/9 < 1.77\bar{9}$. □

4 Robustnes Analysis

In this section, we study the robustness of online square-packing algorithms. We first present a lower bound on the robustness of the previously proposed algorithm, Aosp of [24]. Aosp forms an offline packing of predicted frequency to reserve a placeholder for *all* non-tiny items. Therefore, Aosp is overly reliant on the correctness of the predictions. In particular, one can generate adversarial inputs, formed only by tiny items, in which all placeholders of Aosp remain empty.

Theorem 7 *The* Aosp *algorithm of [24] has a robustness of at least* 21.

Proof. We show that Aosp has a competitive ratio of $841/40 \approx 21$ on an input sequence σ and adversarial frequency predictions $\hat{\mathbf{f}}$, which implies a lower bound of 21 on its robustness.

Aosp forms an offline packing of predicted frequency to reserve a placeholder for *all* expected items, except for "tiny" items. In the context of the Aosp algorithm, tiny items have a size of at most $1/15$. As a result, Aosp is overly reliant on the correctness of the predictions. e. g., all placeholders of some class t remain empty if items of that type never arrive.

Given a frequency prediction $\hat{\mathbf{f}}$, encoded by an adversarial oracle, predicting $7n$ small items of size at most $1/4$ and n large items of size at most $3/5$, Aosp forms an offline packing of n bins, where each bin contains one placeholder for a large item, seven placeholders for small items, and forty containers of size at most $1/15$ to pack tiny items online.

Consider an input sequence σ_w consist of $40n$ tiny items of size $1/30 + \epsilon$. Given the prediction $\mathbf{f}$, $\textsc{Aosp}(\sigma, \mathbf{f})$ has n bins that are partially filled by tiny items. Opt, however, fits each $29^2 = 841$ of these tiny items into a one bin; therefore, $\textsc{Opt}(\sigma) = 40n/841$. It follows, $\textsc{Aosp}(\sigma, \mathbf{f})$ has a competitive ratio of $841/40 \approx 21$ which completes the proof. □

We now show that Rap has a robustness of at most 5.89. For that, we prove a lower bound for the minimum occupied area of the bins that Rap opens.

Theorem 8 Rap *has a robustness of at most* 5.89.

Proof. The final packing of Rap consists of harmonic, LM, and critical bins. A small harmonic bin of class i has a minimum occupied area of $i^2/(i+1)^2$, which implies a minimum occupied area of $9/16$ for all small harmonic bins. Moreover, the occupied area in bins that include large or medium items is at least $1/4$.

Next, we consider bins with an empty placeholder. Placeholders for these items were reserved during the preprocessing step and remained empty due to prediction errors. We note that these bins have received non-empty containers with small or tiny items; otherwise, they would be virtually open (do not contain any items), and do not contribute to the final cost. Therefore, the occupied area of LM-bins is at least the minimum total occupied area of the small or tiny items they contain. Table 4 in Appendix shows lower bounds for the occupied area. In particular, the worst-case scenario is realized by the $\langle 1b, 9\rangle$ bins, as shown in Figure 1g, when the placeholder for $1b$-items stays empty and the occupied area is at least 0.17. We can conclude that all bins in the final packing of Rap, except possibly a constant number of them, have a minimum occupied area of at least 0.17, regardless of the quality of predictions. It follows, Rap has a robustness of at most $100/17 \lessapprox 5.89$. □

5 Concluding Remarks

In this paper, we introduced Rap, an online square-packing algorithm that leverages frequency predictions and has superior consistency and robustness over an existing algorithm of [24], Aosp, which overly relies on predictions. The techniques we used for the design of Rap, e.g., differentiating between placeholders and containers, and its analysis, e.g., solving the integer program in Lemma 5, are likely helpful in studying other geometric packing problems under the prediction model such as 2-dimensional box packing [16, 27] and d-dimensional cube packing [17, 21] problems.

References

[1] Algorithms with predictions. https://algorithms-with-predictions.github.io/. Accessed: 2023-02-19.

[2] S. Angelopoulos and S. Kamali. Contract scheduling with predictions. *J. Artif. Intell. Res.*, 77:395–426, 2023.

[3] S. Angelopoulos, C. Dürr, S. Jin, S. Kamali, and M. Renault. Online computation with untrusted advice. In *Proc. ITCS*, pages 52:1–52:15, 2020.

[4] S. Angelopoulos, S. Kamali, and K. Shadkami. Online bin packing with predictions. In *Proc. IJCAI*, pages 4574–4580, 2022.

[5] A. Antoniadis, T. Gouleakis, P. Kleer, and P. Kolev. Secretary and online matching problems with machine learned advice. In *Proc. NeurIPS*, 2020.

[6] S. Assmann, D. Johnson, D. Kleitman, and J.-T. Leung. On a dual version of the one-dimensional bin packing problem. *Journal of Algorithms*, 5(4): 502–525, 1984.

[7] Y. Azar, D. Panigrahi, and N. Touitou. Online graph algorithms with predictions. In *Proc. SODA*, pages 35–66, 2022.

[8] J. Balogh, J. Békési, G. Dósa, L. Epstein, and A. Levin. Lower bounds for several online variants of bin packing. *Theory of Computing Systems*, 63 (8):1757–1780, 2019.

[9] S. Banerjee, V. Cohen-Addad, A., and Z. Li. Graph searching with predictions. In *Proc. ITCS*, volume 251, pages 12:1–12:24, 2023.

[10] N. Bansal, J. R. Correa, C. Kenyon, and M. Sviridenko. Bin packing in multiple dimensions: Inapproximability results and approximation schemes. *Mathematics of Operations Research*, 31(1):31 – 49, 2006.

[11] J. Boyar, L. M. Favrholdt, and K. S. Larsen. Online unit profit knapsack with untrusted predictions. In *Proc. SWAT*, pages 20:1–20:17, 2022.

[12] J. Boyar, L. M. Favrholdt, S. Kamali, and K. S. Larsen. Online interval scheduling with predictions. In *Proc. WADS*, 2023.

[13] C. L. Canonne. A short note on learning discrete distributions, 2020. arXiv math.ST:2002.11457.

[14] J. Y. Chen, T. Eden, P. Indyk, H. Lin, S. Narayanan, R. Rubinfeld, S. Silwal, T. Wagner, D. P. Woodruff, and M. Zhang. Triangle and four cycle counting with predictions in graph streams. In *Proc. ICLR*, 2022.

[15] J. Y. Chen, S. Silwal, A. Vakilian, and F. Zhang. Faster fundamental graph algorithms via learned predictions. In *Proc. ICML*, volume 162, pages 3583–3602, 2022.

[16] F. R. K. Chung, M. R. Garey, and D. S. Johnson. On packing two-dimensional bins. *SIAM Journal on Algebraic and Discrete Methods*, 3:66–76, 1982.

[17] J. Csirik and G. J. Woeginger. On-line packing and covering problems. In *Online Algorithms, The State of the Art*, volume 1442 of *LNCS*, pages 147–177. Springer, 1996.

[18] S. Dobrev, R. Královič, and D. Pardubská. Measuring the problem-relevant information in input. *RAIRO - Theor. Inf. Appl.*, 43(3):585–613, 2009.

[19] F. Eberle, A. Lindermayr, N. Megow, L. Nölke, and J. Schlöter. Robustification of online graph exploration methods. In *Proc. AAAI*, pages 9732–9740, 2022.

[20] L. Epstein and L. Mualem. Online bin packing of squares and cubes. In *Algorithms and Data Structures*, pages 357–370. Springer International Publishing, 2021.

[21] L. Epstein and L. Mualem. Online bin packing of squares and cubes. In *WADS*, volume 12808, pages 357–370. Springer, 2021.

[22] L. Epstein and R. van Stee. Optimal online algorithms for multidimensional packing problems. *SIAM Journal on Computing*, 35(2):431–448, 2005.

[23] S. Im, R. Kumar, M. M. Qaem, and M. Purohit. Online knapsack with frequency predictions. In *Proc. NuerIPS*, pages 2733–2743, 2021.

[24] S. Kamali and A. López-Ortiz. Almost online square packing. In *Proc. CCCG*, 2014.

[25] T. Lavastida, B. Moseley, R. Ravi, and C. Xu. Learnable and instance-robust predictions for online matching, flows and load balancing. In *Proc. ESA*, volume 204, pages 59:1–59:17, 2021.

[26] T. Lavastida, B. Moseley, R. Ravi, and C. Xu. Using predicted weights for ad delivery. In *Proc. ACDA*, pages 21–31, 2021.

[27] A. Lodi, S. Martello, and M. Monaci. Two-dimensional packing problems: A survey. *Eur. J. Oper. Res.*, 141(2):241–252, 2002.

[28] T. Lykouris and S. Vassilvitskii. Competitive caching with machine learned advice. In *Proc. ICML*, pages 3302–3311, 2018.

[29] M. Mitzenmacher and S. Vassilvitskii. Algorithms with predictions. In T. Roughgarden, editor, *Beyond the Worst-Case Analysis of Algorithms*, pages 646–662. Cambridge University Press, 2020.

[30] M. Purohit, Z. Svitkina, and R. Kumar. Improving online algorithms via ML predictions. In *Proc. NeurIPS*, pages 9661–9670, 2018.

[31] D. Rohatgi. Near-optimal bounds for online caching with machine learned advice. In *Proc. SODA*, pages 1834–1845, 2020.

[32] A. Wei and F. Zhang. Optimal robustness-consistency trade-offs for learning-augmented online algorithms. In *Proc. NeurIPS*, 2020.

[33] A. Zeynali, B. Sun, M. Hajiesmaili, and A. Wierman. Data-driven competitive algorithms for online knapsack and set cover. In *Proc. AAAI*, pages 10833–10841, 2021.

Appendix (Omitted Tables)

	reserved types					
small type	$1b$	$1c$	$1d$	$1e$	$1e, 4$	—
3	—	0.67	0.61	**0.56**	—	0.56
4	—	0.64	**0.58**	**0.53***	—	0.64
5	0.69	0.60	0.74	0.69	0.66	0.69
6	0.66	0.76	0.71	0.65	0.67	0.73
7	0.64	0.73	0.67	0.76	0.67	0.76
8	0.62	0.70	0.78	0.73	0.66	0.79
9	**0.61**	0.81	0.75	0.81	0.64	0.81
10	0.74	0.78	0.83	0.77	0.72	0.82
11	0.72	0.75	0.80	0.75	0.72	0.84
12	0.70	0.83	0.77	0.81	0.71	0.85
13	0.68	0.80	0.83	0.78	0.69	0.86
14	0.67	0.78	0.81	0.83	0.68	0.87
15	0.76	0.84	0.86	0.81	0.75	0.87
16	0.74	0.82	0.84	0.85	0.74	0.88
17	0.73	0.80	0.82	0.83	0.72	0.89
18	0.71	0.85	0.86	0.87	0.72	0.89
19	0.70	0.83	0.84	0.84	0.71	0.90
20	0.77	0.82	0.88	0.88	0.76	0.90
21	0.75	0.86	0.86	0.86	0.74	0.91
22	0.74	0.84	0.84	0.84	0.74	0.91
23	0.73	0.83	0.88	0.87	0.73	0.91
24	0.72	0.87	0.86	0.85	0.73	0.92
25	0.77	0.85	0.89	0.88	0.75	0.92
26	0.76	0.84	0.87	0.86	0.75	0.92
27	0.75	0.87	0.86	0.89	0.75	0.92
28	0.74	0.86	0.89	0.87	0.74	0.93
29	0.73	0.84	0.87	0.90	0.73	0.93
tiny	0.70	**0.61**	0.75	0.70	0.67	0.71

Table 2: A summary of the minimum occupied area in each LM bin type with no empty containers, rounded to 2 decimal places. For each large type, the minimum occupied area is highlighted. The entry marked with * shows the minimum occupied area of critical bins in the final packing.

Class	t_{i+1}	weight
$1a$	4/5	1.0
$1b$	2/3	0.765625
$1c$	3/5	0.6785570840932904
$1d$	11/20	0.4650876739816575
$1e$	1/2	0.4650876739816575
$2a$	2/5	0.25
$2b$	1/3	0.19140625
3	1/4	0.1111111111111111
4	1/5	0.0764160465740489
5	1/6	0.04
6	1/7	0.0277777777777777
7	1/8	0.0222880135840976
8	1/9	0.015625
9	1/10	0.0137867647058823
10	1/11	0.01
11	1/12	0.0082644628099173
12	1/13	0.0069444444444444
13	1/14	0.0059171597633136
14	1/15	0.0051020408163265
15	1/16	0.0044444444444444
16	1/17	0.00390625
17	1/18	0.0034602076124567
18	1/19	0.0030864197530864
19	1/20	0.002770083102493
20	1/21	0.0025
21	1/22	0.0022675736961451
22	1/23	0.0020661157024793
23	1/24	0.0018903591682419
24	1/25	0.0017361111111111
25	1/26	0.0016
26	1/27	0.0014792899408284
27	1/28	0.0013717421124828
28	1/29	0.0012755102040816
29	1/30	0.0011890606420927
tiny density		1.2500557840816486

Table 3: Weights of items of different classes, as used in Lemma 5

	reserved types							
small	1b		1c		1d		1e	
types	containers	total area	containers	total area	containers	total area	containers	total area
3	—	—	5	0.3125	5	0.3125	5	0.3125
4	—	—	7	0.2800	7	0.2800	7	0.2800
5	9	0.2500	9	0.2500	16	0.4444	16	0.4444
6	11	0.2244	20	0.4081	20	0.4081	20	0.4081
7	13	0.2031	24	0.3750	24	0.3750	33	0.5156
8	15	0.1851	28	0.3456	39	0.4814	39	0.4814
9	17	**0.1700**	45	0.4500	45	0.4500	56	0.5600
10	36	0.2975	51	0.4214	64	0.5289	64	0.5289
11	40	0.2777	57	0.3958	72	0.5000	72	0.5000
12	44	0.2603	80	0.4733	80	0.4733	95	0.5621
13	48	0.2448	88	0.4489	105	0.5357	105	0.5357
14	52	0.2311	96	0.4266	115	0.5111	132	0.5866
15	81	0.3164	125	0.4882	144	0.5625	144	0.5625
16	87	0.3010	135	0.4671	156	0.5397	175	0.6055
17	93	0.2870	145	0.4475	168	0.5185	189	0.5833
18	99	0.2742	180	0.4986	203	0.5623	224	0.6204
19	105	0.2625	192	0.4799	217	0.5424	240	0.5999
20	144	0.3265	204	0.4625	256	0.5804	279	0.6326
21	152	0.3140	245	0.5061	272	0.5619	297	0.6136
22	160	0.3024	259	0.4896	288	0.5444	315	0.5954
23	168	0.2916	273	0.4739	333	0.5781	360	0.6250
24	176	0.2816	320	0.5120	351	0.5615	380	0.6079
25	225	0.3328	336	0.4970	400	0.5917	429	0.6346
26	235	0.3223	352	0.4828	420	0.5761	451	0.6186
27	245	0.3125	405	0.5165	440	0.5612	504	0.6428
28	255	0.3032	423	0.5029	495	0.5885	528	0.6278
29	265	0.2944	441	0.4899	517	0.5744	585	0.6500
tiny	9	0.2571	9	0.2571	16	0.4571	16	0.4571

Table 4: Lower bounds for the area ccupied by tiny and small items in LM bins of Rap. The minimum occupied area over all classes is highlighted.