Online Bin Packing with Predictions *

Spyros Angelopoulos, Shahin Kamali, Kimia Shadkami

Online Bin Packing with Predictions*

Spyros Angelopoulos† Shahin Kamali‡ Kimia Shadkami §

Abstract

Bin packing is a classic optimization problem with a wide range of applications, from load balancing to supply chain management. In this work, we study the online variant of the problem, in which a sequence of items of various sizes must be placed into a minimum number of bins of uniform capacity. The online algorithm is enhanced with a (potentially erroneous) *prediction* concerning the frequency of item sizes in the sequence. We design and analyze online algorithms with efficient tradeoffs between the consistency (i.e., the competitive ratio assuming no prediction error) and the robustness (i.e., the competitive ratio under adversarial error), and whose performance degrades near-optimally as a function of the prediction error. This is the first theoretical and experimental study of online bin packing under competitive analysis, in the realistic setting of learnable predictions. Previous work addressed only extreme cases with respect to the prediction error, and relied on overly powerful and error-free oracles.

1 Introduction

Bin packing is a classic optimization problem and one of the original NP-hard problems (Garey & Johnson, 1979). Given a set of *items*, each with a (positive) *size*, and a bin *capacity*, the objective is to assign the items to the minimum number of bins so that the sum of item sizes in each bin does not exceed the bin capacity. Bin packing is instrumental in modelling resource allocation problems such as load balancing and scheduling (Coffman, Garey, & Johnson, 1996), and has many applications in areas such as supply chain management, e.g., capacity planning in logistics and cutting stock, but also in cloud computing, where a cloud provider must decide how many physical machines are needed in order to accommodate the incoming jobs (Cohen, Keller, Mirrokni, & Zadimoghaddam, 2019). Several approximation algorithms have been proposed, e.g., (de la Vega & Lueker, 1981; Rothvoß, 2013; Hoberg & Rothvoß, 2017), and efficient exact heuristics motivated by AI settings are often used in practice (Korf, 2002, 2003; Fukunaga & Korf, 2007; Schreiber & Korf, 2013).

In this work, we focus on the *online* variant of bin packing, in which the set of items is not known in advance but is rather revealed in the form of a *sequence*. Upon the arrival of a new item, the online algorithm must either place it into one of the currently open bins, as long as this action does not violate the bin's capacity, or into a new bin. The online model has several applications related to dynamic

*A preliminary version of this work appeared in the Proceedings of the 31st International Joint Conference on Artificial Intelligence (Angelopoulos, Kamali, & Shadkami, 2022).

†Sorbonne Université, CNRS, LIP6, Paris, France

‡Department of Electrical Engineering and Computer Science, York University, Toronto, Canada

§Department of Computer Science, University of Manitoba, Winnipeg, Canada

resource management, such as virtual machine placement for server consolidation (Song, Xiao, Chen, & Luo, 2013; Wang, Meng, & Zhang, 2011) and memory allocation in data centers (Bein, Bein, & Venigella, 2011). Online bin packing has a long history of study; in Section 1.2 we discuss, in more detail, some of the most significant known results in this setting.

In order to analyze the performance of an online algorithm, we will rely on the well-established framework of *competitive analysis*, which provides strict, theoretical performance guarantees against worst-case scenarios. In fact, as stated in (Coffman et al., 1996), bin packing has served as "an early proving ground for this type of analysis in the broader context of online computation". The *competitive ratio* of an online algorithm is defined as the worst-case ratio of the algorithm's cost (total number of opened bins) over the optimal offline cost (optimal number of opened bins given knowledge of all items). For bin packing, in particular, the standard performance metric is the *asymptotic competitive ratio*, in which the optimal offline cost is arbitrarily large (Coffman et al., 1996).

While the standard online framework assumes that the algorithm has no information on the input sequence, a recently emerged and very active direction in Machine Learning seeks to leverage *predictions* on the input. More precisely, the algorithm has access to some machine-learned information on the input, which, however, may be erroneous; namely, there is a *prediction error* η associated with it. The objective is to design algorithms which perform well if the prediction is accurate, maintain an efficient competitive ratio is the prediction is highly erroneous (i.e., adversarial), and also exhibit a gentle degradation of the competitive performance, as a function of the prediction error. Following the influential work (Lykouris & Vassilvitskii, 2021), we refer to the competitive ratio of an algorithm with an error-free prediction as the *consistency* of the algorithm, and to the competitive ratio with an adversarial prediction as its *robustness*. Several online optimization problems have been studied in this learning-augmented setting, including caching (Lykouris & Vassilvitskii, 2021; Rohatgi, 2020), ski rental and non-clairvoyant scheduling (Purohit, Svitkina, & Kumar, 2018; Wei & Zhang, 2020), makespan scheduling (Lattanzi, Lavastida, Moseley, & Vassilvitskii, 2020), rent-or-buy problems (Banerjee, 2020; Anand, Ge, & Panigrahi, 2020; Gollapudi & Panigrahi, 2019), secretary and matching problems (Antoniadis, Gouleakis, Kleer, & Kolev, 2020b; Lavastida, Moseley, Ravi, & Xu, 2020), and metrical task systems (Antoniadis, Coester, Eliás, Polak, & Simon, 2020a). This is only a partial list of some representative results; see also the survey (Mitzenmacher & Vassilvitskii, 2020).

1.1 Contribution

We give the first theoretical and experimental study of online bin packing with machine-learned predictions. Previous work on this problem has assumed ideal and error-free predictions that must be provided by a very powerful oracle, without any learnability considerations, as we discuss in more detail in Section 1.2. In contrast, our algorithms exploit natural, and PAC-learnable predictions concerning the *frequency* at which item sizes occur in the input, and our analysis incorporates the prediction error into the performance guarantee. As in other AI-motivated works on bin packing, namely (Korf, 2002, 2003; Fukunaga & Korf, 2007; Schreiber & Korf, 2013), we assume a discrete model in which item sizes are integers in $[1, k]$ for some constant k (see Section 2). This assumption is not indispensable, and in Section 5.2 we extend the analysis to items that may have fractional sizes.

We first present and analyze an algorithm called PROFILEPACKING, that achieves optimal consistency, and is also efficient if the prediction error is relatively small. The algorithm builds on the concept of a *profile set*, which serves as an approximation of the items that are expected to appear in the se-

quence, given the frequency predictions. This is a natural concept that, perhaps surprisingly, has not been exploited in the long history of competitive analysis of bin packing, and which can be readily applicable to other online packing problems, such as multi-dimensional packing (Christensen, Khan, Pokutta, & Tetali, 2017) and vector packing (Azar, Cohen, Kamara, & Shepherd, 2013), as we discuss in Section 7.

As the prediction error grows, PROFILEPACKING may not be robust; we show, however, that this is an unavoidable price that any optimally-consistent algorithm with frequency predictions must pay. We thus design and analyze a more general class of *hybrid* algorithms that combine PROFILEPACKING and any one of the known robust online algorithms, and which offers a more balanced theoretical tradeoff between robustness and consistency.

We perform extensive experiments on our algorithms. Specifically, we evaluate them on a variety of publicly available benchmarks, such as the BPPLIB benchmarks (Delorme, Iori, & Martello, 2018), but also on distributions studied specifically in the context of offline bin packing, such as the *Weibull* distribution (Castiñeiras, Cauwer, & O'Sullivan, 2012). The results show that our algorithms outperform the known efficient algorithms without any predictions. We also evaluate a heuristic that updates the predictions based on previously served items, and which is better suited for inputs generated from distributions that change over time (e.g., in the case of *evolving data* (Gomes, Barddal, Enembreck, & Bifet, 2017)).

Last, we show that our algorithms can be applicable in other settings. Specifically, we show an application of our algorithms in the context of *Virtual Machine* (VM) placement in large data centers (Mann, 2015): here, we obtain a more refined competitive analysis in terms of the *consolidation ratio*, which reflects the maximum number of VMs per physical machine, in typical scenarios. Furthermore, we show that our analysis of PROFILEPACKING has a direct application in the *sampling-based* setting, in which the algorithm can access a small sample of the input, and the objective is to obtain an online algorithm that performs efficiently as a function of the number of sampled input items. Thus, our online algorithms can also serve as fast approximations to the offline problem, since frequency prediction with bounded error can be attained with a small sample size.

In terms of analysis techniques, we note that the theoretical analysis of the algorithms we present is specific to the setting at hand and treats items "collectively". In contrast, almost all known online bin packing algorithms are analyzed using a *weighting* technique (Coffman et al., 1996), which treats each bin "individually" and independently from the others (by assigning weights to items and independently comparing a bin's weight in the online algorithm and the optimal offline solution). In terms of the experimental analysis, in our experiments, the prediction error is a natural byproduct of the learning phase, and predictions are obtained by observing a small prefix of the input sequence. This is in contrast to several works in learning-enhanced algorithms, in which a perfect prediction is first generated by a very powerful oracle, then some random error is applied in order to simulate the imperfect prediction.

1.2 Related Work

Online bin packing has a long history of study. The simplest algorithm is NEXTFIT, which places an item into its single open bin when possible; otherwise, it closes the bin (does not use it anymore) and opens a new bin for the item. FIRSTFIT is another simple heuristic that places an item into the first bin of sufficient space and opens a new bin if required. BESTFIT works similarly, except that it places the item into the bin of minimum available capacity, which can still fit the item. NEXTFIT has

a competitive ratio of 2, while both FIRSTFIT and BESTFIT are 1.7-competitive (Coffman et al., 1996; Johnson, Demers, Ullman, Garey, & Graham, 1974). Improving upon this performance requires more sophisticated algorithms, and many have been proposed in the literature. These algorithms are variants of the classic *Harmonic* algorithm (Lee & Lee, 1985), which places items of approximately equal sizes, according to a harmonic sequence, in the same bin. The currently best algorithm is the Advanced Harmonic (AH) algorithm, which has a competitive ratio of 1.57829 (Balogh, Békési, Dósa, Epstein, & Levin, 2018), whereas the best-known lower bound on the competitive ratio is 1.54278 (Balogh, Békési, Dósa, Epstein, & Levin, 2021). However, one should note that the Harmonic family of algorithms are designed specifically for improving the competitive ratio, and their typical performance is inferior to heuristics that are widely used in practice such as FIRSTFIT and BESTFIT, as shown in (Kamali & López-Ortiz, 2015a).

Beyond competitive analysis, the bin packing problem has been studied under stochastic settings, in which the sizes of items are independent and identically distributed samples from some distribution (Csirik, Johnson, Kenyon, Orlin, Shor, & Weber, 2006a; Rhee & Talagrand, 1993; Gupta & Radovanovic, 2020). In this setting, the objective is to minimize the expected *loss*, defined as the difference between the number of bins opened by the algorithm, and the total size of all items normalized by the bin capacity. Ideally, one aims for a loss that is as small as $o(n)$, where n is the number of items in the sequence. For example, the Sum-of-Squares (SS) algorithm of (Csirik et al., 2006a) has an expected loss of $O(\sqrt{n})$. The same guarantee can be achieved by an algorithm whose actions only depend on the size of the arriving item, and the levels of the bins in the current packing (Gupta & Radovanovic, 2020). The algorithms in the above works are oblivious to the distribution; however, if the length of the input n and the distribution are known to the algorithm, the expected loss can be reduced to $O(1)$ (Banerjee & Freund, 2020). We note, however, that all these algorithms are designed for stochastic inputs, and do not provide efficient worst-case guarantees on the competitive ratio. For example, the Sum-of-Square algorithm has a competitive ratio in the range $[2, 2.\bar{7}]$ (Csirik et al., 2006a), which is as bad as (and possibly worse than) the naive NEXTFIT algorithm.

Online bin packing has also been studied under the *advice complexity* model (Boyar, Kamali, Larsen, & López-Ortiz, 2016; Mikkelsen, 2016; Angelopoulos, Dürr, Kamali, Renault, & Rosén, 2018), in which the online algorithm has access to some error-free information on the input called *advice*. The objective is to quantify the tradeoffs between the competitive ratio and the size of the advice (i.e., the number of bits in the binary encoding of the advice). For instance, (Angelopoulos et al., 2018) showed that $O(1)$ advice bits suffice to improve the competitive ratio of the problem to $1.5 + \epsilon$, for any constant $\epsilon > 0$. We emphasize, however, that such studies are only of theoretical interest for two reasons: First, the advice is assumed to have no errors, and it is possible that a single bit flip gravely affects the competitive ratio; Second, the advice is assumed to encode *any* information, without any learnability considerations, and it may thus be provided by an omnipotent oracle that knows the optimal solution. To illustrate with an example, typical advice in the above works encodes information about the number of bins in the optimal offline packing that contain relatively large items (of size at least a third of the bin capacity) or relatively small items (less than a third of the bin capacity).

Online bin packing was recently studied under an extension of the advice complexity model, in which the advice may be *untrusted* (Angelopoulos, Dürr, Jin, Kamali, & Renault, 2020). Here, the algorithm's performance is evaluated only at the extreme cases in which the advice is either error-free or adversarially generated, namely with respect to its consistency and its robustness, respectively. The objective is to find Pareto-efficient algorithms concerning these two metrics, as a function of the advice

size. However, this model is not concerned with the algorithm's performance in typical cases in which the prediction does not fall in one of the two above extremes, does not incorporate the prediction error into the analysis, and does not consider the learnability aspects of the advice. In particular, even with error-free predictions, the algorithm of (Angelopoulos et al., 2020) has a competitive ratio as large as 1.5, whereas a single bit error may result in a competitive ratio that is as large as 6.

Concerning the application of frequency predictions in competitive online optimization, we note that, perhaps surprisingly, such predictions have not been used widely, despite their simplicity and effectiveness. (Mahdian, Nazerzadeh, & Saberi, 2007) gave improved competitive ratios for a generalized online matching problem motivated by advertisement space allocation, using unreliable frequency estimates of keywords. Recently, and concurrently with the conference version of our work, (Im, Kumar, Qaem, & Purohit, 2021) studied the online knapsack problem under frequency predictions, where each item has a value and a size, and the objective is to maximize the value of items that are accepted (and can fit) in the knapsack. Here, the concept of "frequency prediction" has a more liberal notion: it describes, for each possible value, an estimate of the total size of all items of that value. In contrast, in the bin packing setting, item frequency is a less complicated concept, which benefits the design and applicability of the corresponding algorithms.

2 Online Bin Packing: Model and Predictions

We begin with some preliminary discussions related to online bin packing. The input to the online algorithm is a sequence $\sigma = a_1, \ldots, a_n$, where a_i is the size of the i-th item in σ. We denote by n the length of σ, and by $\sigma[i, j]$ the subsequence of σ that consists of items with indices $i, \ldots, j$ in σ.

We denote by $k \in \mathbb{Z}^+$ the bin capacity. Note that k is independent of n, and is thus constant. We assume that the size of each item is an integer in $[1, k]$, where k is the bin capacity. This is a natural assumption on which many efficient algorithms for bin packing rely, e.g., (Schreiber & Korf, 2013; Fukunaga & Korf, 2007; Csirik, Johnson, Kenyon, Orlin, Shor, & Weber, 2006b). Furthermore, without any restriction on the item sizes, (Mikkelsen, 2016) showed that no online algorithm with advice of size sublinear in the size of the input can have competitive ratio better than 1.17 (even if the advice is error-free). This negative result implies that some restriction on item sizes is required so as to leverage frequency predictions. Nevertheless, in Section 5.2 we show how to handle fractional items, and how to express the performance loss due to such items.

Given an online algorithm A (with no predictions), we denote by $A(\sigma)$ its output on input σ, i.e., the packing it produces, and by $|A(\sigma)|$ the number of bins in its output. We denote by $\textsc{Opt}(\sigma)$ the offline optimal algorithm with knowledge of the input sequence. The (asymptotic) competitive ratio of A is defined (Coffman et al., 1996) as

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

Consider a bin b. For the purpose of the analysis, we will often associate b with a specific configuration of items that can be placed into it. We thus say that b is of *type* $(s_1, s_2, \ldots, s_l, e)$, with $s_i \in [1, k]$, $e \in [0, k]$ and $\sum_{j=1}^{l} s_j + e = k$, in the sense that the bin can pack l items of sizes $s_1, \ldots, s_l$, with a remaining empty space equal to e. We specify that a bin is *filled according to type* $(s_1, s_2, \ldots, s_l, e)$, if it contains l items of sizes $s_1, \ldots, s_l$, with an empty space e. Note that a type induces a partition of k into $l + 1$ integers; we call each of the l elements $s_1, \ldots, s_l$ a *placeholder*, and denote by τ_k the number

of all possible bin types. Observe that τ_k depends only on k and not on the length n of the sequence, and is constant, since k is constant.

Consider an input sequence σ. For any $x \in [1, k]$, let $n_{x,\sigma}$ denote the number of items of size x in σ. We define the *frequency of size x in σ*, denoted by $f_{x,\sigma}$, to be equal to $n_{x,\sigma}/n$, hence $f_{x,\sigma} \in [0, 1]$, and $\sum_{x\in[1,k]} f_{x,\sigma} = 1$. Our algorithms will use size frequencies as predictions. Namely, for every $x \in [1, k]$, there is a predicted value of the frequency of size x in σ, which we denote by $f'_{x,\sigma}$. The predictions come with an error, and in general, $f'_{x,\sigma} \neq f_{x,\sigma}$; for instance, it may very well be the case that $\sum_{x\in[1,k]} f'_{x,\sigma} \neq 1$. To quantify the prediction error, let $\boldsymbol{f_\sigma}$ and $\boldsymbol{f'_\sigma}$ denote the frequencies and their predictions in σ, respectively, as points in the k-dimensional space. In line with previous work on online algorithms with predictions, e.g. (Purohit et al., 2018), we can define the error η as the L_1 norm of the distance between $\boldsymbol{f_\sigma}$ and $\boldsymbol{f'_\sigma}$. These size frequencies are PAC-learnable, due to the following (folklore) fact:

Remark 1. *(Canonne, 2020). for any given $\epsilon > 0$ and $\delta \in (0, 1]$, a sample of size $\Theta((k+\log(1/\delta))/\epsilon^2)$ is sufficient (and necessary) to learn the frequencies of k item sizes with accuracy ϵ and error probability δ, assuming the distance measure is the L_1-distance.*

We denote by $A(\sigma, \boldsymbol{f'_\sigma})$ the output of A on input σ and predictions $\boldsymbol{f'_\sigma}$. To simplify notation, we will omit σ when it is clear from context, i.e., we will use $\boldsymbol{f'}$ in place of $\boldsymbol{f'_\sigma}$.

3 Profile Packing

In this section, we present and analyze an online algorithm with predictions $\boldsymbol{f'}$, which we call PROFILEPACKING. The algorithm is based on the concept of a *profile*, denoted by $P_{\boldsymbol{f'}}$, which we define as the multiset that consists of $\lceil f'_x m \rceil$ items of size x, for all $x \in [1, k]$. Here, m is a parameter that is a sufficiently large, but constant integer, which will be specified later. One may thus think of the profile as an "approximation" of the multiset of items that is expected as input, given the predictions $\boldsymbol{f'}$.

Consider an *optimal* packing of the items in $P_{\boldsymbol{f'}}$. Since the size of items in $P_{\boldsymbol{f'}}$ is bounded by k, it is possible to compute such an optimal packing in constant time, e.g., using an efficient exact heuristic (Korf, 2002)). We will denote by $p_{\boldsymbol{f'}}$ the number of bins in the optimal packing of all items in the profile. Note that each of these $p_{\boldsymbol{f'}}$ bins is filled according to a certain type that is specified by the optimal packing of the profile. We simplify notation and use P and p instead of $P_{\boldsymbol{f'}}$ and $p_{\boldsymbol{f'}}$, respectively, when $\boldsymbol{f'}$ is implied.

We define the actions of PROFILEPACKING. Prior to serving any items, PROFILEPACKING opens p empty bins of types that are in accordance with the optimal packing of the profile (so that there are $\lceil f'_x m \rceil$ placeholders of size x in these empty bins). When an item, say of size x, arrives, the algorithm will place it into any placeholder reserved for items of size x, provided that such one exists. Otherwise, i.e., if all placeholders for size x are occupied, the algorithm will open another set of p bins, again of types determined by the optimal profile packing. We call each such set of p bins a *profile group*. Note that the algorithm does not close any bins at any time, that is, any placeholder for an item of size x can be used at any point in time, so long as it is unoccupied.

We require that PROFILEPACKING opens bins in a *lazy* manner, that is, the p bins in the profile group are opened virtually, and each bin contributes to the cost only after receiving an item. Last, suppose that for some size x, it is $f_x > 0$, whereas its prediction is $f'_x = 0$. In this case, x is not in the profile set P.

We call items of such size *special*. PROFILEPACKING packs these special items separately from others, using FIRSTFIT. Algorithm 1 describes PROFILEPACKING in pseudocode.

Algorithm 1 PROFILEPACKING

Input: σ: the input sequence with items in $[1, k]$
$\boldsymbol{f'}$: predicted item frequencies ($\forall x \in [1, k]$, $f'_x \in [0, 1]$)

Output: a packing of σ (a set of bins that contain all items in σ)

▷ form the profile set.
$P_{\boldsymbol{f'}} \leftarrow \phi$
for $x \in \{1, \dots k\}$ **do**
    $P_{\boldsymbol{f'}} \leftarrow P_{\boldsymbol{f'}} \cup \{\lceil f'_x m \rceil$ items of size $x\}$
end for

▷ compute an optimal packing of the profile set in which each bin is partitioned to placeholders.
$\text{OPT}_{\boldsymbol{f'}}(P) =$ optimal packing of $P_{\boldsymbol{f'}}$.
$p_{f'} \leftarrow |\text{OPT}_{\boldsymbol{f'}}(P)|$
$Group \leftarrow p_{f'}$ empty bins in accordance with $\text{OPT}_{\boldsymbol{f'}}(P)$. ▷ open the first profile group
$Empty \leftarrow Group$ ▷ bins that are opened but empty
$NonEmpty \leftarrow \phi$ ▷ bins that contribute to the cost
for $i \in (1, \dots, n)$ **do** ▷ packing the sequence in an online manner
    $x \leftarrow \sigma[i]$
    if $\text{OPT}_{\boldsymbol{f'}}(P)$ has no placeholder of size x **then**
        use FIRSTFIT to pack $\sigma[i]$ ▷ x is a special item
    else
        $N_x \leftarrow$ bins in $NonEmpty$ with placeholder for x
        if $N_x \neq \phi$ **then** ▷ place $\sigma[i]$ in a non-empty bin
            $B \leftarrow$ any bin of N_x
            place $\sigma[i]$ in a placeholder of size x in B
        else
            $E_x \leftarrow$ bins in $Empty$ with placeholder for x
            if $E_x = \phi$ **then** ▷ open a new profile group
                $Group \leftarrow p_{f'}$ new bins as in $\text{OPT}_{\boldsymbol{f'}}(P)$
                $Empty \leftarrow Empty \cup Group$
            end if
            ▷ place $\sigma[i]$ in a (virtually) opened empty bin
            $B \leftarrow$ any bin of E_x
            place $\sigma[i]$ in a placeholder of size x in B
            $Empty \leftarrow Empty \setminus \{B\}$
            $NonEmpty \leftarrow NonEmpty \cup \{B\}$
        end if
    end if
end for
return $NonEmpty$ ▷ return the set of non-empty bins

Example 1. *Suppose that $k = 10$, $m = 20$, and that the predictions $\boldsymbol{f'}$ are given in the following table.*

x	1	2	3	4	5	6	7	8	9	10
f'_x	0.11	0.53	0.15	0.10	0	0.03	0.05	0	0.03	0

The profile based on these frequencies is $P_{\boldsymbol{f'}} = \{3 \times 1, 11 \times 2, 3 \times 3, 2 \times 4, 6, 7, 9\}$, where $i \times x$ indicates i items of size x. For example, there are $\lceil f'_1 m \rceil = \lceil 0.11 \cdot 20 \rceil = 3$ items of size $x = 1$ in the profile. There is an optimal packing of $P_{\boldsymbol{f'}}$ that consists of 7 bins. Figure 1 illustrates this packing, as well as the packing of PROFILEPACKING *on an example online sequence.*

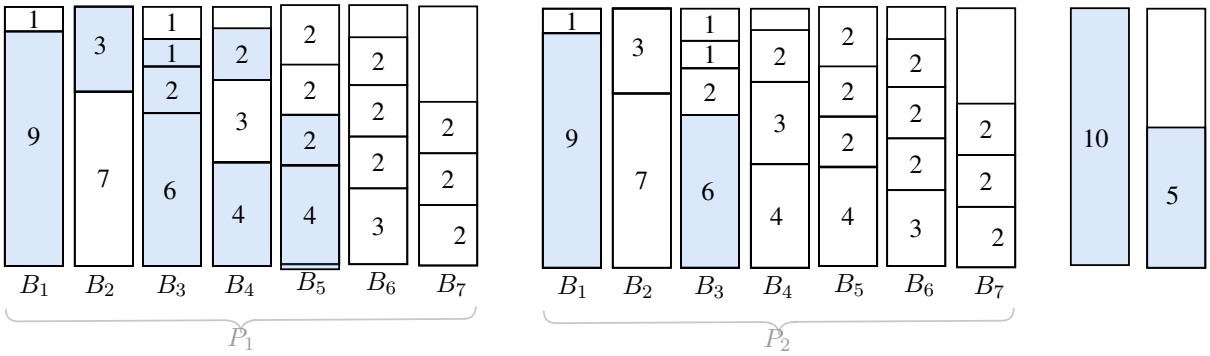


Figure 1: The packing output by PROFILEPACKING on the input sequence $\sigma = 2, 3, 1, 4, 10, 2, 9, 4, 6, 9, 2, 6, 5$. The predictions and the corresponding profile are as given in Example 1. The optimal packing of the profile consists of seven bins $B_1, \ldots, B_7$. When serving σ, the algorithm opens two profile groups, denoted by P_1 and P_2. The profile group P_2 is opened upon serving the second item of size 9, namely the 10th request in the sequence. The placeholders that have received an item are highlighted. Items 10 and 5 are special items and are packed using FIRSTFIT. The total cost for serving σ is equal to 9, and there are 7 bins that are only opened virtually, hence they do not contribute to the cost.

3.1 Analysis of PROFILEPACKING

We first show that in the ideal setting of error-free prediction, PROFILEPACKING is near-optimal (Lemma 2). This result will be very useful in the analysis of the more realistic setting of erroneous predictions (Theorem 3). We denote by ϵ any fixed constant less than 0.5, and in order to achieve consistency equal to $1 + \epsilon$, it will suffice to define m to be such that $m \geq 3\tau_k k/\epsilon$, as will become evident in the proof of Lemma 2. Given that k (and thus τ_k) and ϵ are constants, m is also a constant.

Lemma 2. *For any constant $\epsilon \in (0, 0.5]$, and error-free prediction ($\boldsymbol{f'} = \boldsymbol{f}$),* PROFILEPACKING *has competitive ratio at most $1 + \epsilon$.*

Proof. Given an input sequence σ, denote by $PP(\sigma, \boldsymbol{f'})$ the packing output by the algorithm. This output can be seen as consisting of g profile group packings for some positive integer g (since each time the algorithm allocates a new set of p bins, a new profile group is generated). Since the input consists of n items, and the profile has at least m items, we have that $g \leq \lceil n/m \rceil$.

Given an optimal packing $\textsc{Opt}(\sigma)$, we define a new packing, denoted by N, that not only packs items in σ, but also additional items as follows. N contains all (filled) bins of $\textsc{Opt}(\sigma)$, along with their corresponding items. For every bin type in $\textsc{Opt}(\sigma)$, we want that N contains a number of bins of that type that is divisible by g. To this end, we add up to $g-1$ filled bins of the same type in N.

We can argue that $|N|$ is not much bigger than $|\textsc{Opt}(\sigma)|$. We have that

$$|N| \leq |\textsc{Opt}(\sigma)| + (g-1)\tau_k < |\textsc{Opt}(\sigma)| + n\tau_k/m \leq |\textsc{Opt}(\sigma)|(1+\tau_k k/m),$$

Specifically, the first inequality holds from the way N was generated, the second inequality holds because $g \leq \lceil n/m \rceil$ (thus $g-1 < n/m$), and the last inequality holds because $|\textsc{Opt}(\sigma)| \geq \lceil n/k \rceil$ (each bin can contain at most k items). Let $\epsilon' = \epsilon/3$ and recall that we have chosen m so that $m \geq \tau_k k/\epsilon'$. We conclude that

$$|N| \leq (1+\epsilon')|\textsc{Opt}(\sigma)|. \tag{1}$$

By construction, N contains g copies of the same bin (i.e., bins that are filled according to the same type). Equivalently, N consists of g copies of the same packing, which we denote by $\overline{N}$. Define $q = |\overline{N}|$ to be the number of bins in this packing. We will show that p is not much bigger than q, which is crucial in the proof. The number of items of size x in the packing $\overline{N}$ is at least $\lceil n_x/g \rceil$, since N contains at least n_x items of size x. We can give the following lower bound on $\lceil n_x/g \rceil$.

$$\begin{aligned}
\lceil n_x/g \rceil &\geq n_x/\lceil n/m \rceil && (g \leq \lceil n/m \rceil)\\
&> n_x m/(n+m) && (\lceil n/m \rceil < (n+m)/m)\\
&= n_x(m/n - m^2/(n^2+mn))\\
&\geq n_x m/n - m^2/(n+m) && (n_x \leq n)\\
&\geq \lceil n_x m/n \rceil - 1 - m^2/(n+m) && (y \geq \lceil y \rceil - 1 \text{ for any } y)\\
&> \lceil n_x m/n \rceil - 2. && (m^2 < n)
\end{aligned}$$

The last inequality holds because m is a constant with respect to n, which defines the input size. We conclude, from the above, that $\lceil n_x/g \rceil \geq \lceil n_x m/n \rceil - 1$. Given that there are $\lceil f'_x m \rceil = \lceil n_x m/n \rceil$ items of size x in the profile set, we can further conclude that for any $x \in [1,k]$, $\overline{N}$ packs all items of size x that appears in the profile set, with the exception of at most one such item. From the statement of PROFILEPACKING, and its optimal packing of the profile set, we infer that

$$q + k \geq p, \tag{2}$$

and recall that p denotes the size of the optimal packing of the profile. Moreover, we have:

$$\begin{aligned}
q = |\overline{N}| &= |N|/g && \text{(by definition of } q\text{)} \\
&\geq |\textsc{Opt}(\sigma)|/g \geq n/(kg) && (|N| \geq |\textsc{Opt}(\sigma)| \geq n/k) \\
&\geq n/(k\lceil n/m \rceil) && (g \leq \lceil n/m \rceil) \\
&> (\lceil n/m \rceil m - m)/(k\lceil n/m \rceil) \\
&\geq m/k - m^2/kn \\
&> m/k - \epsilon' && (m^2/kn \in o(1) \text{ and } \epsilon' \in \Theta(1)) \\
&> \tau_k/\epsilon' - \epsilon' && (m \geq \tau_k k/\epsilon') \\
&> (\tau_k - 1)/\epsilon' > k/\epsilon'. && (\epsilon' < 1/\epsilon', \tau_k > k+1)
\end{aligned}$$

We thus showed that $q > k/\epsilon'$, hence (2) implies that

$$p < q(1+\epsilon'). \tag{3}$$

We conclude that the number of bins in each profile group is within a factor $(1+\epsilon')$ of the number of bins in $\overline{N}$. Moreover, recall that $PP(\sigma, \boldsymbol{f'})$ consists of g profile groups, and N consists of g copies of $\overline{N}$. Combining this with previously shown properties, we have that

$$\begin{aligned}
|PP(\sigma, \boldsymbol{f'})| &\leq g \cdot p \\
&< g(1+\epsilon')q && \text{(by Inequality (3))} \\
&\leq (1+\epsilon')(1+\epsilon')|\textsc{Opt}(\sigma)| && \text{(by Inequality (1))} \\
&< (1+3\epsilon')|\textsc{Opt}(\sigma)| && ((1+\epsilon')^2 < 1+3\epsilon') \\
&= (1+\epsilon)|\textsc{Opt}(\sigma)|,
\end{aligned}$$

which concludes the proof. □

We will now use Lemma 2 to prove a more general result that incorporates the prediction error into the analysis. To this end, we will relate the cost of the packing of ProfilePacking to the packing that the algorithm would output if the prediction were error-free, which will allow us to apply the result of Lemma 2. Specifically, we will argue that in the presence of prediction error, the cost of ProfilePacking may be affected in two ways: The number of bins in a single profile of ProfilePacking may increase, and more profiles may have to be opened. In the proof of the following theorem, for each of these two cases, we bound the number of additional opened bins as a function of error.

Theorem 3. *For any constant $\epsilon \in (0, 0.5]$, and predictions $\boldsymbol{f'}$ with error η,* ProfilePacking *has competitive ratio at most $1 + (2+5\epsilon)\eta k + \epsilon$.*

Proof. Let $\boldsymbol{f}$ be the frequency vector for the input σ, where $\boldsymbol{f}$ is unknown to the algorithm. In this context, $PP(\sigma, \boldsymbol{f})$ is the packing output by ProfilePacking with error-free prediction, and from Lemma 2 we know that $|PP(\sigma, \boldsymbol{f})| \leq (1+\epsilon)|\textsc{Opt}(\sigma)|$. Recall that $P_{\boldsymbol{f'}}$ denotes the profile set of ProfilePacking on input σ with predictions $\boldsymbol{f'}$, and $p_{\boldsymbol{f'}}$ denotes the number of bins in the optimal packing of $P_{\boldsymbol{f'}}$; $P_{\boldsymbol{f}}$ and $p_{\boldsymbol{f}}$ are defined similarly. We will first relate $p_{\boldsymbol{f}}$ and $p_{\boldsymbol{f'}}$ in terms of the error η. Note that the multisets $P_{\boldsymbol{f}}$ and $P_{\boldsymbol{f'}}$ differ in at most $\sum_{x=1}^{k} \mu_x$ elements, where $\mu_x = |\lceil f_x m \rceil - \lceil f'_x m \rceil|$.

We call these elements *differing*. We have $\mu_x \leq |(f_x - f'_x)m| + 1$, hence $\sum_{x=1}^{k} \mu_x \leq k + \sum_{x=1}^{k} |(f_x - f'_x)m| \leq k + \eta m$, where the last inequality holds because η is the L_1 norm of the distance between $\boldsymbol{f_\sigma}$ and $\boldsymbol{f'_\sigma}$, that is $\eta = \sum_{x=1}^{k} |(f_x - f'_x)|$. We conclude that the number of bins in the optimal packing of $P_{\boldsymbol{f'}}$ exceeds the number of bins in the optimal packing of $P_{\boldsymbol{f}}$ by at most $k+\eta m$, i.e., $p_{\boldsymbol{f'}} \leq p_{\boldsymbol{f}} + k + \eta m$.

Let g and g' denote the number of profile groups in $PP(\sigma, \boldsymbol{f})$ and $PP(\sigma, \boldsymbol{f'})$, respectively. We aim to bound $|PP(\sigma, \boldsymbol{f'})|$, and to this end we will first show a bound on the number of bins opened by $PP(\sigma, \boldsymbol{f'})$ in its first g profile groups, then in on the number of bins in its remaining $g' - g$ profile groups (if $g' \leq g$, there is no such contribution to the total cost). For the first part, the bound follows easily: There are g profile groups, each one consisting of $p_{\boldsymbol{f'}}$ bins, therefore the number of bins in question is at most $g \cdot p_{\boldsymbol{f'}} \leq g(p_{\boldsymbol{f}} + k + \eta m)$. For the second part, since PROFILEPACKING is lazy, any item packed by $PP(\sigma, \boldsymbol{f'})$ in its last $g' - g$ packings has to be a differing element, which implies from the discussion above that $PP(\sigma, \boldsymbol{f'})$ opens at most $g(k + \eta m)$ bins in its last $g' - g$ profile groups. The result follows then from the following inequalities:

$$
\begin{aligned}
&|PP(\sigma, \boldsymbol{f'})| \\
&\leq g(p_{\boldsymbol{f}} + k + \eta m) + g(k + \eta m) \\
&= \ g(p_{\boldsymbol{f}} + 2k + 2\eta m) \\
&\leq \ g(p_{\boldsymbol{f}} + 2\eta m(1+\epsilon)) && (k \leq \epsilon m) \\
&\leq \ g(p_{\boldsymbol{f}} + 2\eta \, p_{\boldsymbol{f}} k(1+\epsilon)) && (p_{\boldsymbol{f}} \geq \lceil m/k \rceil) \\
&= g \cdot p_{\boldsymbol{f}}(1 + 2\eta k(1+\epsilon)) \\
&\leq |PP(\sigma, \boldsymbol{f})|(1 + 2\eta k(1+\epsilon)) \\
&\leq (1+\epsilon)(1 + 2\eta k(1+\epsilon))|\textsc{Opt}(\sigma)| && (\text{since } |PP(\sigma, \boldsymbol{f})| \leq (1+\epsilon)|\textsc{Opt}(\sigma)|) \\
&= (1 + 2\eta k(1+\epsilon)^2 + \epsilon)|\textsc{Opt}(\sigma)| \\
&= (1 + 2\eta k(1 + \epsilon^2 + 2\epsilon) + \epsilon)|\textsc{Opt}(\sigma)| \\
&< (1 + \eta k(2 + 5\epsilon) + \epsilon)|\textsc{Opt}(\sigma)|. && (\epsilon^2 < \epsilon/2)
\end{aligned}
$$

□

Theorem 3 shows that PROFILEPACKING has robustness that is linear in k. The following result proves that this is an unavoidable price that *any* online algorithm with frequency-based predictions must pay.

Theorem 4. *Let c be any constant with $c < 1$. Then for any $\alpha \leq c/k$, any algorithm with frequency predictions that is $(1+\alpha)$-consistent must have robustness at least $(1-c)k/2$.*

Proof. For simplicity, we can assume that n is divisible by k. Suppose that the prediction indicates that in the input sequence σ, half of the items have size 1, while the remaining items have size $k-1$, i.e., we have $f'_{x,\sigma} = 1/2$, if $x \in \{1, k-1\}$, and $f'_{x,\sigma} = 0$, otherwise. Define σ_1 as the sequence that consists of n items of size 1 followed by n items of size $k-1$, and σ_2 as the sequence that consists of n items of size 1 followed by n items of size 1.

Suppose first that the input is σ_1, then the above-defined prediction is error-free. Moreover, $\textsc{Opt}(\sigma_1) = n$ and hence for an algorithm A to be $(1+\alpha)$-consistent, it must open at most $(1+\alpha)n + o(n)$ bins. Out of these bins, n bins each receive an item of size $k-1$ and possibly an item of size 1. Each of the

remaining $\alpha n + o(n)$ bins may contain up to k items of size 1; that is, they contain at most $k\alpha n + o(n)$ items of size 1. Therefore, $n - k\alpha n - o(n)$ items of size 1 must each be placed together with items of size $k-1$. This implies that A opens at least $(1-k\alpha)n - o(n)$ bins when serving the first n items of size 1.

Next, suppose that the input is σ_2. We have $f_{1,\sigma_2} = 1$ and $f_{k-1,\sigma_2} = 0$, and consequently the error is $\eta = 1$. The optimal packing is formed by $2n/k$ bins, each containing k items of size 1; that is, $\textsc{Opt}(\sigma_2) = 2n/k$. On the other hand, by the above observation, the cost of A is at least $(1-k\alpha)n - o(n)$. Hence, the robustness of A is at least $\frac{(1-k\alpha)n-o(n)}{2n/k}$. Given that $\alpha \leq c/k$, it follows that A has robustness at least $(1-c)k/2$. □

We conclude that the robustness of PROFILEPACKING is close-to-optimal and no $(1+\epsilon)$-consistent algorithm can do asymptotically better. It is possible, however, to obtain more general tradeoffs between consistency and robustness, as we discuss in the next section.

Time complexity of PROFILEPACKING We bound the overall time complexity of PROFILEPACKING for serving a sequence of n items as a function of n, k, and m. The initial phase of the algorithm, which involves computing the profile and its optimal packing, runs in time independent of n and does not impact the asymptotic time complexity. It is possible to find the optimal packing of the profile set using efficient exact heuristics such as (Fukunaga & Korf, 2007; Schreiber & Korf, 2013). If faster preprocessing is required, one can replace the exact optimal packing with an approximate packing using simple heuristics like FIRSTFITDECREASING (Dósa, 2007), which has a competitive ratio of $11/9$. This will improve the empirical running time, while increasing the number of opened bins by the same ratio. Such an approach is also useful in settings where predictions are updated based on previously served items, and thus the packing of the profile set must be computed periodically. Overall, the worst-case time complexity of PROFILEPACKING is $O(kmn)$. Note that each item is served in amortized time $O(km)$, which is constant since k and m are constants.

4 A Broader Class of Algorithms

In this section, we describe and analyze a more general class of algorithms which offer better robustness in comparison to PROFILEPACKING, at the expense of slightly worse consistency. To this end, we will combine PROFILEPACKING with any algorithm A that has efficient worst-case competitive ratio, in the standard online model in which there is no prediction. Specifically, we will define a class of algorithms based on a parameter $\lambda \in [0,1]$ and the robust algorithm A, and which we denote by HYBRID(λ); we will also say that HYBRID(λ) is *based* on A. Let $a, b \in \mathbb{N}^+$ be such that $\lambda = a/(a+b)$. We require that the parameter m in the statement of PROFILEPACKING is a sufficiently large constant, namely

$$m \geq 5\tau_k \max\{k, a+b\}/\epsilon, \tag{4}$$

as it will become clear in the proof of Theorem 5.

Upon arrival of an item of size $x \in [1,k]$, HYBRID(λ) marks it as either an item to be served by PROFILEPACKING, or as an item to be served by A; we call such an item a *PP-item* or an *A-item*, in accordance to this action. Moreover, for every $x \in [1,k]$, HYBRID(λ) maintains two counters:

$\texttt{count}(x)$, which is the number of items of size x that have been served so far, and $\texttt{ppcount}(x)$, which is the number of PP-items of size x that have been served so far.

We describe the actions of HYBRID(λ). Suppose that an item of size x arrives. If there is an empty placeholder of size x in a non-empty bin, then the item is assigned to that bin (and to the corresponding placeholder), and declared PP-item. Otherwise, there are two possibilities: If $\texttt{ppcount}(x) \leq \lambda \cdot \texttt{count}(x)$, then it is served using PROFILEPACKING and is declared PP-item. If $\texttt{ppcount}(x) > \lambda \cdot \texttt{count}(x)$, then it is served using A and declared A-item. Thus, λ is a measure of the "involvement" of the two algorithms in serving a given sequence. For extreme values of λ, HYBRID(λ) reduces to one of its two components: if $\lambda = 1$, then HYBRID(λ) reduces to PROFILEPACKING, whereas if $\lambda = 0$, it reduces to A.

Note that in HYBRID(λ), A and PROFILEPACKING maintain their own bin space, so when serving according to one of these algorithms, only the bins opened by the corresponding algorithm are considered. Thus, we can partition the bins used by HYBRID(λ) into *PP-bins* and *A-bins*.

Example 2. *Suppose that $k = 10$, $m = 20$, and that the predictions $\boldsymbol{f'}$ are as in Example 1. Figure 2 illustrates the packing of the* HYBRID*(λ) algorithm that is based on* FIRSTFIT *as algorithm A, with a parameter $\lambda = 0.5$.*

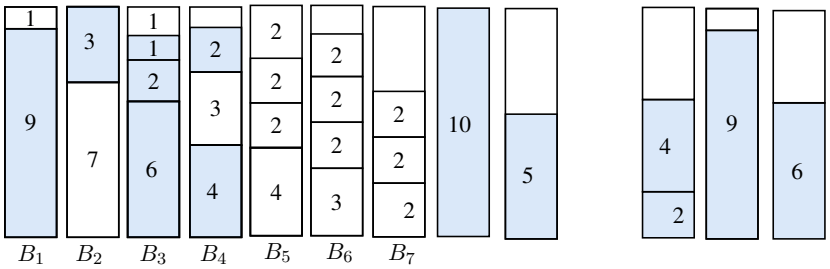


Figure 2: The packing of HYBRID(λ) on the input sequence $\sigma = 2, 3, 1, 4, 10, 2, 9, 4, 6, 9, 2, 6, 5$. The predictions and profile are described in Example 1. The total cost incurred by the algorithm on σ is equal to 9, with PROFILEPACKING and FIRSTFIT contributing 6 and 3 bins, respectively.

Before presenting the analysis of HYBRID(λ), we review two other approaches towards robustifying PROFILEPACKING based on a given (robust) algorithm A, and we show that these approaches do not perform as well as HYBRID(λ). Recall that HYBRID(λ) will first place an item x to a placeholder of size x if such a placeholder is available. The first approach could be to skip this step; that is, consider an algorithm that serves a fraction λ of items of size x as PP-items, and the remaining $1 - \lambda$ fraction as A-items. To exemplify the problem with this approach, suppose that k is divisible by 5, and consider a situation where the prediction specifies that half of the items have size $0.6k$, and the other half have size $0.4k$. Then, PROFILEPACKING will use two placeholders per profile bin, one of size $0.6k$ and another of size $0.4k$ in the optimal packing of the profile set. Suppose that the input sequence σ of length n, and it consists only of items of size $0.4k$ and $0.6k$, in non-decreasing order of size. Moreover, the total frequency of items of size $0.4k$ is equal to $0.5 + \eta/2$, whereas the total frequency of items of size $0.6k$ is equal to $0.5 - \eta/2$, for some $\eta > 0$ (hence η is the prediction error). Then, the above algorithm opens

$\lambda n/2$ bins for PP-items, each including an item of size $0.4k$ and a placeholder of size $0.6k$. Given that the initial check for placeholders is skipped, only $\lambda n(0.5 - \eta/2)$ item of size $0.6k$ are placed in these placeholders. That is, in the final packing of the algorithm, there are $\lambda\eta n/2$ bins with a single item of size $0.4k$ and an empty placeholder of size $0.6k$. HYBRID(λ), on the other hand, places an item of size $0.6k$ in each of these bins, and thus saves $\lambda\eta n/4$ bins.

A second approach could be along the lines of (Mahdian, Nazerzadeh, & Saberi, 2012), which describe a general method for combining an *optimistic* algorithm that trusts the prediction (in our context, PROFILEPACKING) and a *pessimistic* algorithm that ignores the prediction (in our context, the online algorithm A). The optimistic and pessimistic algorithms optimize for situations where the prediction is perfect and adversarial, respectively. (Mahdian et al., 2012) showed that their combined algorithm attains the best of both worlds performance for problems such as load balancing and facility location. Their algorithm serves an input item using the optimistic one, if the optimistic algorithm would have incurred a total cost that would be bounded by a constant factor of the corresponding cost that would be incurred by the pessimistic algorithm, up to that point in time; otherwise, it serves the item using the pessimistic algorithm. In our context, this approach would treat an item as PP-item if the total cost of PROFILEPACKING on the prefix of the input observed so far is within a constant factor $\alpha > 0$ of the cost of A on the same prefix. Unfortunately, this approach fails for the bin packing problem. Consider a worst-case example, where half of the items are predicted to be of size $0.4k$, and the remaining half are predicted to be of size $0.6k$. Suppose that this prediction is error-free and all items of size $0.4k$ appear before items of size $0.6k$ in the sequence. Then PROFILEPACKING reserves place-holders of size $0.6k$ in its bins, and its cost on any prefix of the input formed by items of size $0.4k$ is exactly twice as large as the cost of A for on the same prefix: this is because A must place two such items in the same bin, because of its pessimistic nature. We observe that if $\alpha \leq 2$, this combined algorithm treats all items of size $0.4k$ as A-items, and its final packing will be the same as the one of A. Similarly, if $\alpha > 2$, the algorithm treats all items as PP-items and thus reduces to PROFILEPACKING. In other words, this approach fails to meaningfully combine the two algorithms.

4.1 Analysis of HYBRID(λ)

The following theorem establishes the performance of HYBRID(λ).

Theorem 5. *For any $\epsilon \in (0, 0.5]$ and $\lambda \in [0, 1]$,* HYBRID*(λ) has competitive ratio $(1+\epsilon)((1+(2+5\epsilon)\eta k+\epsilon)\lambda + c_A(1-\lambda))$, where c_A is the competitive ratio of the algorithm A on which* HYBRID*(λ) is based.*

Proof. We define two partitions of the multiset of items in σ. The first partition is $S_{PP} \cup S_A$, where S_{PP} and S_A are the PP-items and A-items of HYBRID(λ), respectively. The second partition is $S'_{PP} \cup S'_A$, where S'_{PP} and S'_A are defined such that for any $x \in [1, k]$ there are $\lfloor \lambda n_x \rfloor$ items of size x in S'_{PP} and $n_x - \lfloor \lambda n_x \rfloor$ items of size x in S'_A. Given OPT(σ), we will define a new packing N, such that every bin in N contains only items in S'_{PP} or only items in S'_A. Let N_{PP} and N_A denote the set of bins in N that include items in S'_{PP} and in S'_A, respectively. Similarly, let B_{PP} and B_A denote the set of bins in the packing of HYBRID(λ) that contain only PP-items (PP-bins) and A-items (A-bins), respectively. We will prove the following bounds for N, B_{PP} and B_A:

(i) $|N| \leq (1+\epsilon)|\text{OPT}(\sigma)|$;

(ii) $|B_{PP}| \leq (1 + (2 + 5\epsilon)\eta k + \epsilon)|N_{PP}|$;

(iii) $|B_A| \leq c_A|N_A|$.

To prove the above bounds, we first explain how to derive N from a given optimal packing $\textsc{Opt}(\sigma)$. This is done in a way that N contains the filled bins of $\textsc{Opt}(\sigma)$ and up to $(a + b)\tau_k$ additional filled bins so as to guarantee that, for each bin type in $\textsc{Opt}(\sigma)$, the number of bins of each given type in N is divisible by $a + b$; recall that a, b are the smallest integers such that $\lambda = \frac{a}{a+b}$. Using (4), we obtain that

$$|N| \leq |\textsc{Opt}(\sigma)| + (a + b - 1)\tau_k < |\textsc{Opt}(\sigma)|(1 + \tau_k k/m),$$

Since the number of bins of each type in N is divisible by $a + b$, we can partition N into N_{PP} and N_A so that $|N_{PP}| \leq a(1 + \epsilon)|\textsc{Opt}(\sigma)|/(a + b)$ and $|N_A| \leq b(1 + \epsilon)|\textsc{Opt}(\sigma)|/(a + b)$. That is, $|N_{PP}| \leq \lambda(1 + \epsilon)|\textsc{Opt}(\sigma)|$ and $|N_A| \leq (1 - \lambda)(1 + \epsilon)|\textsc{Opt}(\sigma)|$. Note that N packs not only items in σ but also additional items in the added bins. That implies that all items S'_{PP} are packed in N_{PP} and all items in S'_A are packed in N_A, and hence (i) follows.

To prove (ii) and (iii), we note that $S_A \subseteq S'_A$ which implies $S'_{PP} \subseteq S_{PP}$. This is because the algorithm declares an item of size x as an A-item only if $\texttt{ppcount}(x) > \lambda\ \texttt{count}(x)$. Hence, at any given time during the execution of $\textsc{Hybrid}(\lambda)$, the number of A-items of size x is no more than a fraction $(1 - \lambda)$ of $\texttt{count}(x)$.

Next, we will show the property (ii). First, note that $|B_{PP}| = |PP(\sigma_{PP}, \boldsymbol{f'})|$, where σ_{PP} is the subsequence of σ formed by the PP-items, and PP abbreviates the output of PROFILEPACKING. Consider a sequence σ'_{PP} obtained by removing, for every $x \in [1, k]$, the last d_x items of size x from σ_{PP}, where d_x is the number of items of size x in $S_{PP} \setminus S'_{PP}$. We show next that $|PP(\sigma_{PP}, \boldsymbol{f'})| = |PP(\sigma'_{PP}, \boldsymbol{f'})|$. For any x, consider the last PP-item L_x of size x for which $\textsc{Hybrid}(\lambda)$ opens a new bin. At the time L_x is packed, $\texttt{ppcount}(x) \leq \lambda \cdot \texttt{count}(x)$. Thus, by removing items of size x that appear after L_x in σ_{PP}, the remaining items form a subsequence of σ'_{PP}, and the number of bins does not decrease. That implies that $|PP(\sigma_{PP}, \boldsymbol{f'})| = |PP(\sigma'_{PP}, \boldsymbol{f'})|$. From Theorem 3, we obtain

$$\begin{aligned}
|B_{PP}| &= |PP(\sigma_{PP}, \boldsymbol{f'})| \\
&= |PP(\sigma'_{PP}, \boldsymbol{f'})| \\
&\leq (1 + (2 + 5\epsilon)\eta k + \epsilon)|\textsc{Opt}(\sigma'_{PP})| \\
&\leq (1 + (2 + 5\epsilon)\eta k + \epsilon)|N_{PP}|.
\end{aligned}$$

Last, to show (iii), we note that the number of bins that $\textsc{Hybrid}(\lambda)$ opens for items in S_A is at most $c_A|\textsc{Opt}(S_A)| \leq c_A|\textsc{Opt}(S'_A)| \leq c_A|N_A|$. This is because $S_A \subseteq S'_A$.

Using Properties (i)-(iii), we obtain

$$\begin{aligned}
|Hybrid(\sigma, \boldsymbol{f'})| = &\ |B_{PP}| + |B_A| \\
\leq &\ (1 + (2 + 5\epsilon)\eta k + \epsilon)|N_{PP}| + c_A|N_A| \\
\leq &\ (1 + (2 + 5\epsilon)\eta k + \epsilon)\lambda(1 + \epsilon)|\textsc{Opt}(\sigma)| + c_A(1 - \lambda)(1 + \epsilon)|\textsc{Opt}(\sigma)| \\
= &\ (1 + \epsilon)((1 + (2 + 5\epsilon)\eta k + \epsilon)\lambda + c_A(1 - \lambda))|\textsc{Opt}(\sigma)|,
\end{aligned}$$

which concludes the proof. □

To obtain the best theoretical performance, we can choose A as the algorithm of the best known competitive ratio, that is Advanced Harmonic algorithm (Balogh et al., 2018). However, as discussed in Section 2, such algorithms belong to a class that is tailored to worst-case competitive analysis, and do not tend to perform well in typical instances (Kamali & López-Ortiz, 2015b). For this reason, simple algorithms such as FIRSTFIT and BESTFIT are preferred in practice (Coffman et al., 1996). We obtain the following corollary.

Corollary 6. *For any $\epsilon \in (0, 0.5]$ and $\lambda \in [0, 1]$, there is an algorithm with competitive ratio $(1 + \epsilon)(1.5783+\lambda((2+5\epsilon)\eta k-0.5783+\epsilon))$. Furthermore,* HYBRID*($\lambda$) based on* FIRSTFIT *has competitive ratio $(1 + \epsilon)(1.7 + \lambda((2 + 5\epsilon)\eta k - 0.7 + \epsilon))$.*

We can do even better if an *upper bound* estimation of the error is known to the algorithm. Such an upper bound, which we denote by H, may be available depending on the quality of historical data and the characteristics of typical sequences. Specifically, let H-AWARE denote the algorithm which executes HYBRID(1), if $H < (c_A - 1 - \epsilon)/(k(2 + 5\epsilon))$, and HYBRID(0), otherwise. An equivalent statement is that H-AWARE executes PROFILEPACKING if $H < (c_A - 1 - \epsilon)/(k(2 + 5\epsilon))$, and A, otherwise. The following corollary follows directly from Theorem 5 with the observation that as long as $\eta < (c_A - 1 - \epsilon)/(k(2 + 5\epsilon))$, PROFILEPACKING has a competitive ratio that is better than c_A.

Corollary 7. *For any $\epsilon \in (0, 0.5]$, H-*AWARE *using algorithm A has competitive ratio $\min\{c_A, 1+(2+5\epsilon)\eta k + \epsilon\}$, where c_A is the competitive ratio of A. In particular, choosing* FIRSTFIT *as A, H-*AWARE *has competitive ratio $\min\{1.7, 1 + (2 + 5\epsilon)\eta k + \epsilon\}$.*

5 Applications & Extensions

In this section, we discuss extensions and further applications of our algorithms.

5.1 Virtual Machine Placement

An important application of online bin packing is Virtual Machine (VM) placement in large data centers. Here, each VM corresponds to an item whose size represents the resource requirement of the VM, and each bin corresponds to a physical machine (i.e., host) of a given capacity k. In the context of this application, the *consolidation ratio* (Mann, 2015) is the number of VMs per host, in typical scenarios. Note that the consolidation ratio is typically much smaller than k. For example, VMware server virtualization achieves a consolidation ratio of up to 15:1 (VMware, 2021), while Intel's virtualization infrastructure gives a consolidation ratio of up to 20:1 (Intel, 2010).

Let r denote the consolidation ratio (but note that this quantity is an integer). The following result shows that we can express the competitive ratio of HYBRID(λ) in Theorem 5 so that the capacity k is replaced by the consolidation ratio r. We can thus exploit the fact that typically r is much smaller than k, and improve the theoretical analysis of our algorithms.

Theorem 8. *Consider an instance of online bin packing with bins of capacity k, in which the item sizes are such that at most r items can fit into a bin, for some $r \leq k$. Then, for any constant $\epsilon \in (0, 0.2]$, and predictions $\boldsymbol{f'}$ with error η, the following hold:* (I) PROFILEPACKING *has competitive ratio at most $1 + (2 + 5\epsilon)\eta r + \epsilon$; and* (II) *for any $\lambda \in [0, 1]$,* HYBRID*(λ) has competitive ratio $(1 + \epsilon)((1 + (2 +$*

$5\epsilon)\eta r + \epsilon)\lambda + c_A(1-\lambda))$, *where c_A is the competitive ratio of the algorithm A on which* HYBRID(λ) *is based.*

Proof. The proof of (I) is identical to that of Theorem 3, except that in the fourth inequality that bounds $|PP(\sigma, \mathbf{f}')|$, we use the fact that $p_{\boldsymbol{f}} \geq \lceil m/r \rceil$ (instead of $p_{\boldsymbol{f}} \geq \lceil m/k \rceil$), given that at most r items can fit into each bin. Moreover, in all subsequent inequalities in the proof, k is replaced with r. The proof of (II) is identical to that of Theorem 5, except that part (ii) of Theorem 5 is replaced by $|B_{PP}| \leq (1 + (2 + 5\epsilon)\eta r + \epsilon)|N_{PP}|$, which directly follows from the same arguments and by applying part (I) instead of Theorem 3. □

Similarly, we can generalize Theorem 4 and obtain the following improved impossibility result. The proof is identical to that of Theorem 4, with k replaced by r.

Theorem 9. *Consider an instance of online bin packing with bins of capacity k, in which the item sizes are such that at most r items can fit into a bin, for some $r \leq k$. Then, for any constant $c < 1$, and for any $\alpha \leq c/r$, any algorithm with frequency predictions that is $(1+\alpha)$-consistent has robustness at least $(1-c)r/2$.*

5.2 Handling Items with Fractional Sizes

As stated in Section 2, we assume a discrete model in which items have integral sizes in $[1, k]$. While this is a natural model for many AI applications, our algorithms can also handle *fractional* item sizes in $[1, k]$, by treating them as "special" items, in the sense that they are not predicted to appear. PROFILEPACKING and HYBRID(λ) will then pack these fractional items separately from all integral ones, using FIRSTFIT. For the analysis in this setting, we need a measure of "deviation" of the input sequence σ (that may contain fractional items) from a sequence of integral sizes. The first, and perhaps most natural, approach is to define this deviation as the L_1 distance between σ, and the sequence in which each item is rounded to the closest integer in $[0, k]$. However, we show that this definition can be overly restrictive.

Theorem 10. *Let $\lfloor x \rceil$ denote the integer closest to x, and define $d(\sigma) = \sum_{x \in \sigma} |x - \lfloor x \rceil|$. Then no online algorithm in the fractional setting can have a competitive ratio better than 4/3, even if all frequency predictions are error-free (that is, $\eta = 0$), and even if $d(\sigma) = \epsilon$, for arbitrarily small $\epsilon > 0$.*

Proof. Let $\sigma = \sigma_1\sigma_2$, where σ_1 consists of n items of size $0.5 - \epsilon/(2n)$, and σ_2 consists of n items of size $0.5 + \epsilon/(2n)$. For simplicity, we assume that n and k are even integers. Suppose also that $\boldsymbol{f'}$ is such that $f_{x,\lfloor\sigma\rceil} = 1$, if $x = k/2$, and 0, otherwise (i.e., only items of size $k/2$ are predicted to appear in σ). From the definition of error, it also follows that $\eta = 0$, and from the definition of the deviation d, we have that $d(\sigma) = \epsilon$.

Let A be any online algorithm, then from the definition of σ, we have that $A(\sigma_1, \boldsymbol{f'}) = cn$, for some $c \geq 1/2$. Given that OPT$(\sigma_1) = n/2$, the competitive ratio of A is at least $2c$. Out of the cn bins of $A(\sigma_1, \boldsymbol{f'})$, $n - cn$ bins must have two items, whereas the remaining $cn - (n - cn) = 2cn - n$ bins must have one item. Any of these remaining bins can each accommodate another item from σ_2. Therefore, out of the n items in σ_2, A can pack at most $2cn - n$ such items in the cn bins opened for σ_1, and it must place the remaining $n - (2cn - n) = 2n - 2cn$ items in separate (new) bins. It follows that $A(\sigma, \boldsymbol{f'}) \geq cn + (2n - 2cn) = 2n - cn$. Given that OPT$(\sigma) = n$, the competitive ratio of A is therefore at least $2 - c$. In summary, the competitive ratio of A is $\max\{2c, 2 - c\}$, which is minimized at $4/3$ for $c = 2/3$. □

In light of the above negative result, a different measure of "deviation" can be defined the ratio between the total size of fractional items in σ over the total size of all items in σ. The following theorem shows that this measure can better capture the performance of the algorithm in the fractional setting.

Theorem 11. *Define* $\hat{d}(\sigma) = \frac{\sum_{x\in\sigma, x\neq\lfloor x\rfloor} x}{\sum_{x\in\sigma} x}$. *Let* A *be any algorithm with frequency predictions that has competitive ratio* c *if all items have integral size. Then there is an algorithm* A' *that has competitive ratio at most* $c + 2\hat{d}(\sigma)$ *for inputs with fractional sizes.*

Proof. Let σ_I and σ_F be the subsequences of σ formed by integer and fractional items, respectively. We can write $A(\sigma) = A(\sigma_I) + FF(\sigma_F)$, where $FF(\sigma_F)$ denotes the number of bins opened by FIRSTFIT when serving σ_F. For the number of bins opened for integer items, we have $A(\sigma_I) \leq A(\sigma) \leq c\cdot\textsc{Opt}(\sigma)$. Let $S(\sigma)$ and $S(\sigma_F)$ denote the total size of items in σ and σ_F, respectively, that is $S(\sigma) = \sum_{x\in\sigma} x$, and $S(\sigma_F) = \sum_{x\in\sigma, x\neq\lfloor x\rfloor} x$. From definition, we have $\hat{d}(\sigma) = S(\sigma_F)/S(\sigma)$. Note that $FF(\sigma_F) \leq 2S(\sigma_F)/k + 1$; this is because any pair of consecutive bins contains items of total size $k/2$ or larger. Therefore,

$$FF(\sigma_F) \leq 2\hat{d}(\sigma)S(\sigma)/k + 1 \leq 2\hat{d}(\sigma)\textsc{Opt}(\sigma) + 1.$$

In summary, we have $A(\sigma) \leq c \cdot \textsc{Opt}(\sigma) + 2\hat{d}(\sigma)\textsc{Opt}(\sigma) + 1$, therefore the (asymptotic) competitive ratio of A is at most $c + 2\hat{d}(\sigma)$. □

Example 3. *Suppose that the prediction specifies that half of the items are of size 4 and the remaining half are of size 6. Suppose also that the input* σ *consists of* $n/2$ *items of size 4,* $9n/20$ *items of size 6, and* $n/20$ *items of size* 6.1. *Then,* $\hat{d}(\sigma) = \frac{\frac{n}{20}\cdot 6.1}{\frac{n}{2}\cdot 4 + \frac{9n}{20}\cdot 6 + \frac{n}{20}\cdot 6.1} = \frac{6.1}{101.1} < 0.061$. *Theorem 11 shows that the worst-case, asymptotic competitive ratio of the algorithm cannot exceed* $c + 0.0122$ *in the fractional setting.*

5.3 A Sampling-based Algorithm for Online Bin Packing

Our analysis of PROFILEPACKING, as stated in Theorem 3, in conjunction with the PAC-learnability of frequency predictions, can help obtain a *sampling-based* algorithm with an efficient tradeoff between the number of sampled items and its attained competitive ratio. More precisely, consider the setting in which the online algorithm is allowed to observe s items of the request sequence, and we would like to express its (asymptotic) competitive ratio as a function of s. Similar types of sampling-based competitive analysis have recently attracted attention in the context of other online problems such as ski rental and prophet inequalities (Diakonikolas, Kontonis, Tzamos, Vakilian, & Zarifis, 2021), matching (Kaplan, Naori, & Raz, 2022), and network optimization problems (Argue, Frieze, Gupta, & Seiler, 2022).

Given any small constant $\epsilon > 0$, define $\delta = 1/\sqrt{2^{s\epsilon^2-k}}$. Let ON* denote the best online algorithm in the standard setting, which is currently the Advanced Harmonic algorithm (Balogh et al., 2018) with competitive ratio 1.5783. We define RANDOM-MIX to be the algorithm that works as follows: With probability δ, RANDOM-MIX executes ON^*, whereas, with probability $1-\delta$, it executes PROFILEPACKING. The analysis of this algorithm follows directly from Theorem 3 and Remark 1.

Corollary 12. *For any constant* $\epsilon > 0$ *and* $k \in \mathbb{N}^+$, RANDOM-MIX *with* s *samples has expected competitive ratio* $1.5783\cdot\delta + (1-\delta)(1 + (2+5\epsilon)\eta k + \epsilon)$, *where* $\delta = 1/\sqrt{2^{s\epsilon^2-k}}$.

Note that Corollary 12 bounds the expected competitive ratio of a randomized algorithm which commits to its choice (that is, it executes either ON^* or PROFILEPACKING, and this decision is made once the sample is revealed). In contrast, Corollary 6 expresses the competitive ratio of a deterministic algorithm which judiciously switches between ON^* and PROFILEPACKING throughout its execution, in order to achieve deterministic guarantees.

6 Experimental Evaluation

In this section, we present an experimental evaluation of the performance of our algorithms[1]. Specifically, in Section 6.1 we describe the benchmarks and the input generation model; in Section 6.2, we expand on the predictions and error measurement; and in Section 6.3, we present and discuss the main experimental results. In addition, in Section 6.4 we report further experiments on the profile size, and in Section 6.5 we provide further methodology for reporting the average performance of our algorithms over multiple runs. Last, in Section 6.6, we study the performance of our algorithms in dynamic settings in which the input is generated from an evolving distribution.

6.1 Benchmarks and Input Generation

Several benchmarks have been used in previous work on *offline* bin packing; we refer to the discussion by (Castiñeiras et al., 2012) for a list of related work. Many of these previous benchmarks typically rely on either uniform or normal distributions. There are two important issues to take into account. First, such simple distributions are often unrealistic and do not capture typical applications of bin packing such as resource allocation, as observed in (Gent, 1998). Second, in what concerns online algorithms, simple algorithms such as FIRSTFIT and BESTFIT are very close to optimal for input sequences generated from uniform distributions (Coffman et al., 1996) and very often outperform, in practice, many online algorithms of better competitive ratio (Kamali & López-Ortiz, 2015b).

We evaluate our algorithms on two types of benchmarks. The first type is based on the *Weibull* distribution, which was first proposed in (Castiñeiras et al., 2012) as a model of several real-world applications of bin packing, e.g., the 2012 ROADEF/EURO Challenge on a data center problem provided by Google and several examination timetabling problems. The Weibull distribution is specified by two parameters: the *shape* parameter sh and the *scale* parameter sc (with $sh, sc > 0$). The shape parameter defines the spread of item sizes: lower values indicate greater skew towards smaller items. The scale parameter represents the statistical dispersion of the distribution. In our experiments, we chose $sh \in [1.0, 4.0]$. This is because values outside this range result in trivial sequences with items that are generally too small (hence easy to pack) or too large (for which any online algorithm tends to open a new bin). The scale parameter is not critical, since we scale items to the bin capacity, as we will discuss later; we thus set $sc = 1000$, in accordance with (Castiñeiras et al., 2012).

The second type of benchmarks is generated from the BPPLIB library (Delorme et al., 2018), a collection of bin packing benchmarks used in various works on (offline) algorithms for bin packing. In particular, we report results on the benchmarks "GI" (Gschwind & Irnich, 2016), "Schwerin" (Schwerin & Wäscher, 1997), "Randomly_Generated" (Delorme, Iori, & Martello, 2014), "Schoenfield_Hard28" (Schoenfield, 2002) and "Wäscher" (Wäscher & Gau, 1996).

[1]The code on which the experiments are based is available at https://github.com/shahink84/BinPackingPredictions.

We fix the size of the sequence to $n = 10^6$. We set the bin capacity to $k = 100$, and we also scale down each item to the closest integer in $[1, k]$. This choice is relevant for applications such as Virtual Machine placement (Section 5.1), as explained in Section 5.1. We generate two classes of input sequences. For Weibull benchmarks, the input sequence consists of items generated independently and uniformly at random, and the shape parameter is set to $sh = 3.0$. For BPPLIB benchmarks, we first select a file of the benchmark uniformly at random, then generate input items from the chosen file, again uniformly at random.

6.2 Compared Algorithms, Predictions and Error

We evaluate HYBRID(λ) for $\lambda \in \{0, 0.25, 0.5, 0.75, 1\}$, based on FIRSTFIT. This means that HYBRID(0) is identical to FIRSTFIT, whereas HYBRID(1) is identical to PROFILEPACKING. We fix the size of the profile set to $m = 5000$. To simplify the implementation of PROFILEPACKING, we use the algorithm FIRSTFITDECREASING (Coffman et al., 1996) to compute the profile packing, instead of an optimal algorithm. Specifically, FIRSTFITDECREASING first sorts items in the non-increasing order of their sizes and then packs the sorted sequence using FIRSTFIT. Using FIRSTFITDECREASING helps reduce the time complexity, and the results only improve by using an optimal algorithm for profile packing, instead.

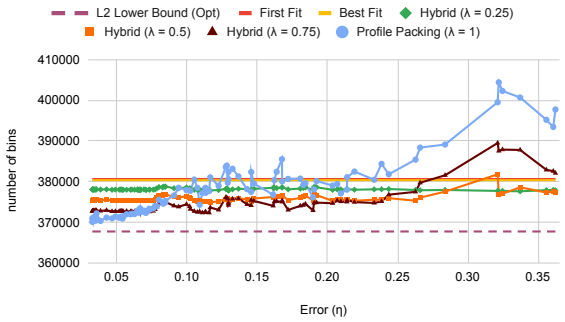
We generate the frequency predictions to HYBRID(λ) as follows: For a parameter $b \in \mathbb{N}^+$, we define the predictions $\boldsymbol{f'}$ as $\boldsymbol{f}_{\boldsymbol{\sigma}[\mathbf{1},\boldsymbol{b}]}$. In words, we use a prefix of size b of the input σ so as to estimate the frequencies of item sizes in σ. In our experiments, we consider 100 different prefix sizes. More precisely, we consider all b of the form $b = \lfloor 100 \cdot 1.05^i \rfloor$, with $i \in [25, 125]$. We define the prediction error η as the L_1 distance between the predicted and the actual frequencies. Note that for a given input sequence, η is a function of the prefix size b. Since we consider 100 distinct values for b, for each sequence we consider up to 100 possible error values. As expected from Remark 1, the prediction error decreases with b.

As explained earlier, FIRSTFIT and BESTFIT perform very well in practice, and we use them as benchmarks for comparing our algorithms. As often in offline bin packing, we also report the *L2 lower bound* (Martello & Toth, 1990; Fukunaga & Korf, 2007) as a lower-bound estimation of the optimal *offline* bin packing solution. That is, no algorithm, online or offline, can perform better than this lower bound.

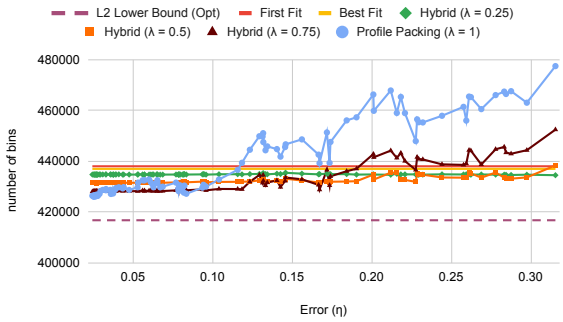
6.3 Results and Discussion

Figure 3 depicts the cost of the algorithms for a typical sequence, as a function of the prediction error. The chosen files are "csBA125_9" (for "GI"), "Schwerin2_BPP32" (for "Shwerin"), "BPP_750_50_0.1_0.8_2" (for "Randomly_Generated"), "Hard28_BPP832" (for "Schoenfield_Hard28"), and "Waescher_TEST0082" (for "Wäscher"). Here, we consider a single sequence, as opposed to averaging over multiple sequences, because each input sequence is associated with its own prediction error, for any given prefix size (and naively averaging over both the cost and the error may produce misleading results). We can use a single sequence because the input size is considerable ($n = 10^6$), and the distribution is fixed. Nevertheless, in Section 6.5 we explain how to properly average over multiple sequences, and we report similar plots and conclusions.

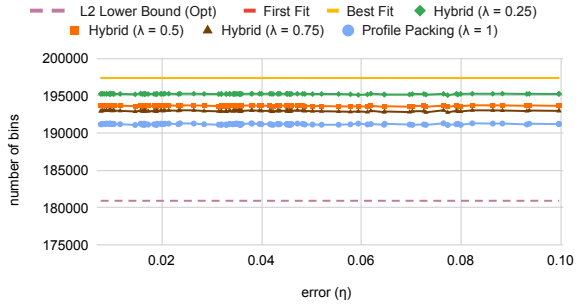
For all benchmarks, we observe that PROFILEPACKING ($\lambda = 1$) degrades quickly as the error



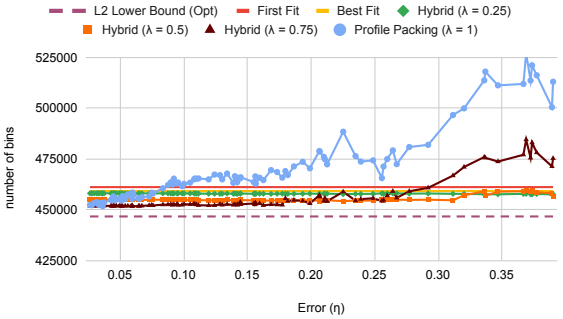
(a) Weibull distribution.



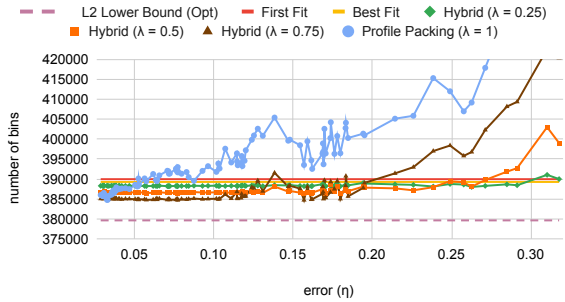
(b) GI benchmark from BPPLIB.



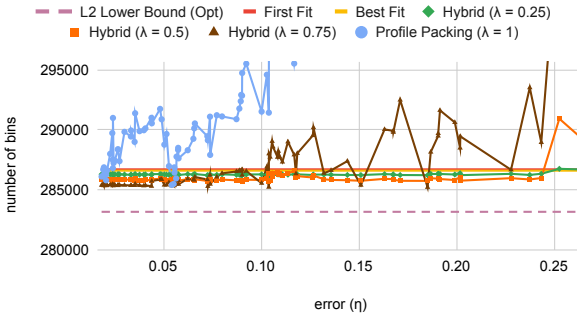
(c) Shwerin benchmark from BPPLIB.



(d) Randomly_Generated benchmark from BPPLIB.



(e) Schoenfield_Hard28 benchmark from BPPLIB.

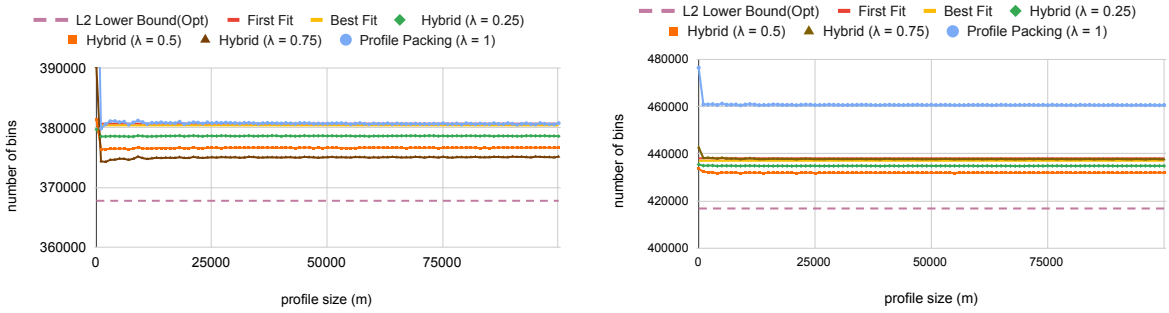


(f) Wäscher benchmark from BPPLIB.

Figure 3: Number of opened bins for sequences from a given distribution. For the purpose of visualization, some of the plots are truncated, e.g., the plot of PROFILEPACKING in (c) and (d).

increases, even though it has very good performance for small values of error. As λ decreases, we observe that HYBRID(λ) becomes less sensitive to error, which confirms the statement of Corollary 6.

Specifically, we observe that for the Weibull benchmarks, HYBRID(λ) dominates both FIRSTFIT and BESTFIT for all $\lambda \in \{0.25, 0.5, 0.75\}$ and for all $\eta < 0.27$, approximately. For the GI benchmarks,



(a) Weibull benchmark.

(b) GI benchmark from BPPLIB.

Figure 4: Number of opened bins as a function of the profile size.

HYBRID(λ) dominates FIRSTFIT and BESTFIT for $\lambda \in \{0.25, 0.5\}$, and for practically all values of error. In the "Shwerin" benchmark, all items have sizes in the range $[15, 20]$. As such, very good predictions can be obtained by observing a tiny part of the input sequence, i.e., for small values of the prefix size b. In particular, the smallest value of b, namely $b = 391$ results in $\eta < 0.099$. As illustrated in Figure 3c, the smaller the parameter λ, the better the performance of HYBRID(λ); in particular, PROFILEPACKING performs the best, which suggests that for inputs from a small set of item sizes, it is beneficial to choose a small value of λ. This can be explained by the fact that the prediction error is relatively smaller for these types of inputs. This finding can be useful in the context of applications such as VM placement: this is because there is only a small number of different VMs that can be assigned to any given physical machine, as we discussed in Section 5.1. For the remaining benchmarks, namely "Randomly_Generated", "Schoenfield_Hard28", and "Wäscher", the relative performance of the algorithms is similar to that for the GI benchmark, with the difference that the divergence of the algorithms becomes observable at different values of the prediction error.

The results demonstrate that frequency-based predictions indeed lead to performance gains. Even for very large prediction error (i.e., a prefix size as small as $b = 338$) HYBRID(λ) with $\lambda \leq 0.5$ outperforms both FIRSTFIT and BESTFIT, therefore the performance improvement comes by only observing a tiny portion of the input sequence.

6.4 Experiments on the Profile Size

In previous experiments, we assumed that the profile size is $m = 5000$. In this section, we report experiments on other values of m. More precisely, we evaluated the performance on two random sequences of length $n = 10^6$ in which the item sizes are generated using Weibull distribution (with $sh = 3$) and the GI-benchmark, respectively, as detailed in Section 6.2. As before, we choose $k = 100$. Predictions are generated based on a prefix of length $b = 1000$ of the input; this resulted in error values of $\eta = 0.1922$ and $\eta = 0.2045$ for the Weibull and GI-instances, respectively. We run HYBRID(λ) ($\lambda \in \{0.25, 0.5, 0.75, 1\}$) for 100 different values of m, equidistant in the interval $[100, 100100]$.

Figure 4 depicts the number of bins opened by the algorithms. The experiments show that the parameter m has little impact on the performance of HYBRID(λ), that is, as long as m is sufficiently

large (e.g., when $m \geq 1000$), the performance of HYBRID(λ) is consistent and independent of the choice of m.

6.5 Experiments on the Average Cost and Rounded Error

In the experiments that we discussed in Section 6.3, we reported the performance of the algorithm on a typical sequence. More precisely, we considered a single randomly generated sequence, as opposed to averaging the cost of the algorithm over multiple input sequences, because each input sequence is associated with its own prediction error, for any given size of the prefix (and averaging naively over both the cost and the error, simultaneously, may produce misleading results). We argued that this should not be an issue, because the input sequence is of considerable size ($n = 10^6$).

In this section, we present further experimental results based on averaging over both the cost and the error which give further justification for this choice. Our setting here is as follows: Given a fixed distribution (either Weibull with $sh = 3$, or a file from the GI Benchmark), we generate 20 random sequences of length 10^6. For each sequence, we compute FIRSTFIT, BESTFIT, and the $L2$ lower bound. The average costs of these algorithms, over the 20 sequences, serve as the benchmark costs for comparison.

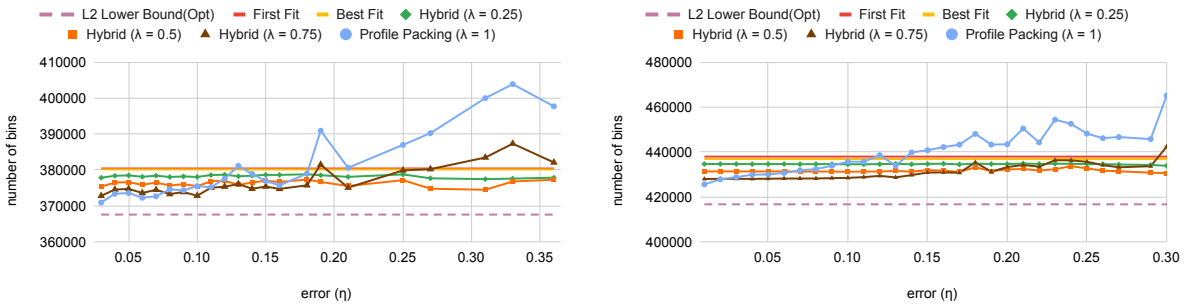
For HYBRID(λ), and every $\lambda \in [0.25, 0.5, 0.75, 1]$, we generate predictions for 100 values of the prefix size b (where recall that b is of the form $b = 100 \cdot 1.05^i$, with $i \in [25, 125]$). Consider a sequence σ. For each of the above predictions for σ, we compute the prediction error as well as the cost of HYBRID(λ) on σ with the corresponding prediction and store a pair of the form (ERROR, COST), where ERROR is the error with a two-digit decimal precision, and the cost is the cost of the algorithm. For example, if ERROR = 0.2341 and COST = 143000, we store the pair (0.23, 143000). This means that for a fixed sequence, we store up to 100 such pairs (assuming $\eta < 1$). Last, we evaluate the average of pairs with the same rounded error over the 20 sequences. For example, if for σ_1 we have obtained the pair (0.23,100000), for σ_2 the pair (0.23, 150000), and for σ_3 the pair (0.23, 350000), then we take the average as the pair (0.23, 200000).

Figure 5 depicts the plots obtained by this method, for both the Weibull and the GI benchmarks. We observe that HYBRID(λ) exhibits similar performance tradeoffs as the plots for a single sequence (Figure 3), but the differences are less pronounced due to averaging.

6.6 Evolving Distributions

In this section, we address the situation in which the input is not drawn according to a fixed distribution but instead is generated from distributions that change with time, e.g., when dealing with evolving data streams. This is a complex setting that has not been studied in any previous work on online bin packing, with or without predictions.

We define a heuristic called ADAPTIVE(w), in which predictions are updated dynamically using a *sliding window* approach; see e.g. (Gomes et al., 2017). ADAPTIVE(w) uses a parameter $w \in \mathbb{N}^+$ as follows. In the initial phase, ADAPTIVE(w) serves $\sigma[1, w]$ using FIRSTFIT; moreover, at the end of this phase, it computes $\boldsymbol{f_{\sigma[1,w]}}$, namely the frequency vector of all sizes in $\sigma[1, w]$. From this point onwards, the algorithm will serve items using PROFILEPACKING with predictions $\boldsymbol{f'}$ which are initialized to $\boldsymbol{f_{\sigma[1,w]}}$. Specifically, every time ADAPTIVE(w) encounters item $\sigma[iw]$, for $i \in \mathbb{N}^+$, it updates $\boldsymbol{f'}$ to $\boldsymbol{f_{\sigma[(i-1)w+1,iw]}}$.



(a) Weibull benchmark.

(b) GI Benchmark from BPPLIB.

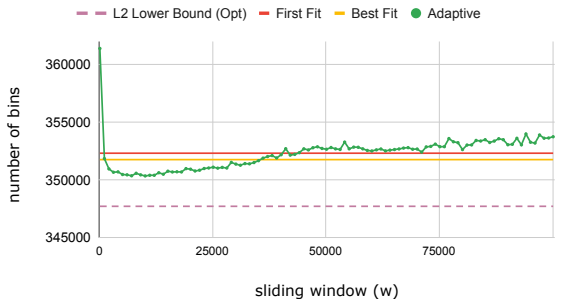
Figure 5: Average number of bins vs. average error over twenty sequences.

For the analysis, we use the benchmarks described in Section 6.1. The distribution of the input sequence changes every 50000 items. Namely, the input sequence is the concatenation of $n/50000$ subsequences. For Weibull benchmarks, each subsequence is a Weibull distribution, whose shape parameter is chosen uniformly at random from $[1.0, 4.0]$. For BPPLIB benchmarks, each subsequence is generated by choosing a file uniformly at random, then generating 50000 items uniformly at random from that specific file.

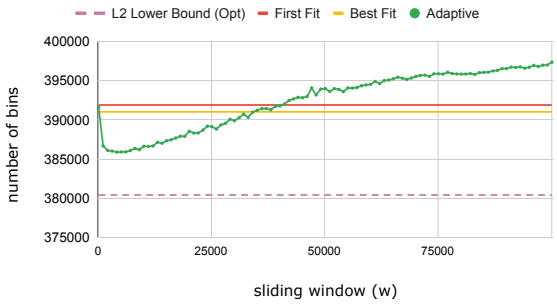
We evaluate ADAPTIVE(w) for 100 values of the sliding window w, equidistant in the range $[100, 100000]$. This is a crucial parameter: if w is too small, we do not obtain sufficient information on the frequencies, whereas if w is too big, the predictions become "stale".

Figure 6 depicts the number of bins opened by ADAPTIVE(w) as a function of w for different benchmarks. Here, we report the average cost of the algorithms over 20 randomly generated sequences. We observe that for the Weibull and "GI" benchmarks, there is a relatively wide range for w that leads to performance improvement, in comparison to FIRSTFIT and BESTFIT, namely it suffices to choose $w \in [2100, 25000]$. For "Randomly_Generated" and "Schoenfield_Hard28", the performance curve of ADAPTIVE(w) is similar to that on the GI benchmark, and ADAPTIVE(w) improves upon FIRSTFIT and BESTFIT when w takes values in the shorter range [2000,4000]. For "Schwerin", ADAPTIVE(w) always performs better, which can be explained by the discussion in Section 6.3. For "Wäscher", ADAPTIVE(w) does not offer any advantage over FIRSTFIT and BESTFIT. However, these two baseline algorithms are remarkably close to the $L2$ lower bound, which means that they output essentially optimal packings for this benchmark, and which in turn leaves very little room for any potential improvement.

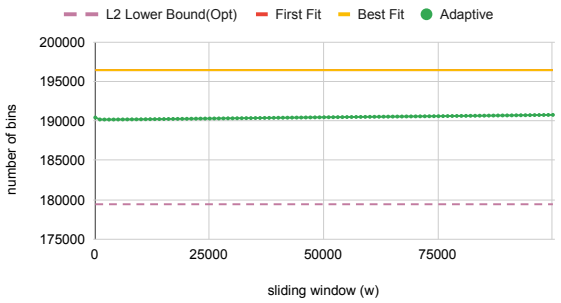
When ADAPTIVE(w) opens a new profile group, the predicted frequencies are updated based on the w most recently packed items. These w items follow a distribution that may have changed since the time a new profile group was opened. As such, the performance of ADAPTIVE(w) depends on the diversity of the distributions that form the benchmark. For example, for "Schwerin", the distribution does not evolve drastically, which explains why ADAPTIVE(w) performs consistently better than FIRSTFIT and BESTFIT, unlike the "Wäscher" benchmark.



(a) Weibull distribution.



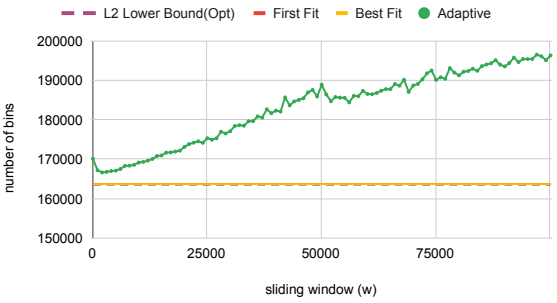
(b) GI benchmark from BPPLIB.



(c) Shwerin benchmark from BPPLIB.

(d) Randomly_Generated benchmark from BPPLIB.

(e) Hard28 benchmark from BPPLIB.



(f) Wäscher benchmark from BPPLIB.

Figure 6: Number of opened bins for sequences from an evolving distribution. For the Shwerin benchmark, FIRSTFIT and BESTFIT open a similar number of bins and they practically coincide in the plot.

7 Conclusion

We gave the first results on the competitive analysis of online bin packing, in a setting in which the algorithm has access to learnable predictions concerning the size frequencies. Our approach exploits the concept of profile packing, which can be applicable in more generalized packing problems, such as two-dimensional setting studied by Chung, Garey, and Johnson (1982) and Huang and Korf (2013) and three-dimensional setting studied by Zhao, She, Zhu, Yang, and Xu (2021) and, more generally, in *vector bin packing* studied by Azar et al. (2013). These are well-studied extensions of the basic online

bin packing problem, with many applications in transportation logistics and cloud computing. In these problems, a main challenge will be to leverage, or develop new offline heuristics for computing the profile packing, since the profile size increases exponentially with the dimension.

Another class of problems for which the approach may be useful is the class of *multicontainer* packing problems, such as multiple knapsack, bin covering, and min-cost covering. For this class of problems, Fukunaga and Korf (2007) gave efficient *bin completion* offline algorithms that can be very useful towards the design of profile-based online algorithms. Last, a further direction for future work on bin packing problems is to incorporate a *distributional* model of predictions, as studied by Diakonikolas et al. (2021), in which the prediction is given as the cumulative distribution function of the item size distribution.

Acknowledgements

This research was supported by the CNRS-Emergence project ONFIN, and by the project PREDICTIONS, grant ANR-19-CE48-0016 from the French National Research Agency (ANR). We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) [funding reference number DGECR-2018-00059].

References

Anand, K., Ge, R., & Panigrahi, D. (2020). Customizing ML predictions for online algorithms. In *International Conference on Machine Learning (ICML)*, pp. 303–313. PMLR.

Angelopoulos, S., Dürr, C., Jin, S., Kamali, S., & Renault, M. P. (2020). Online computation with untrusted advice. In *Proceedings of the 11th Innovations in Theoretical Computer Science Conference (ITCS)*, pp. 52:1–52:15.

Angelopoulos, S., Dürr, C., Kamali, S., Renault, M. P., & Rosén, A. (2018). Online bin packing with advice of small size. *Theory of Computing Systems*, *62*(8), 2006–2034.

Angelopoulos, S., Kamali, S., & Shadkami, K. (2022). Online bin packing with predictions. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 4574–4580. ijcai.org.

Antoniadis, A., Coester, C., Eliás, M., Polak, A., & Simon, B. (2020a). Online metric algorithms with untrusted predictions. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pp. 345–355.

Antoniadis, A., Gouleakis, T., Kleer, P., & Kolev, P. (2020b). Secretary and online matching problems with machine learned advice. In *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS)*.

Argue, C., Frieze, A. M., Gupta, A., & Seiler, C. (2022). Learning from a sample in online algorithms. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS)*.

Azar, Y., Cohen, I. R., Kamara, S., & Shepherd, B. (2013). Tight bounds for online vector bin packing. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 961–970.

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

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

Banerjee, S., & Freund, D. (2020). Uniform loss algorithms for online stochastic decision-making with applications to bin packing. In *Abstracts of the Performance Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, pp. 1–2.

Banerjee, S. (2020). Improving online rent-or-buy algorithms with sequential decision making and ML predictions. In *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS)*.

Bein, D., Bein, W., & Venigella, W. (2011). Cloud storage and online bin packing. In *Proceedings of the 5th International Symposium on Intelligent Distributed Computing (IDC)*, pp. 63–68. Springer.

Boyar, J., Kamali, S., Larsen, K. S., & López-Ortiz, A. (2016). Online bin packing with advice. *Algorithmica*, *74*(1), 507–527.

Canonne, C. L. (2020). A short note on learning discrete distributions.. arXiv math.ST:2002.11457.

Castiñeiras, I., Cauwer, M. D., & O'Sullivan, B. (2012). Weibull-based benchmarks for bin packing. In *Proceedings of the 18th International Conference on Principles and Practice of Constraint Programming (CP)*, Vol. 7514, pp. 207–222.

Christensen, H. I., Khan, A., Pokutta, S., & Tetali, P. (2017). Approximation and online algorithms for multidimensional bin packing: A survey. *Comput. Sci. Rev.*, *24*, 63–79.

Chung, F. R., Garey, M. R., & Johnson, D. S. (1982). On packing two-dimensional bins. *SIAM Journal on Algebraic Discrete Methods*, *3*(1), 66–76.

Coffman, E. G., Garey, M. R., & Johnson, D. S. (1996). Approximation algorithms for bin packing: A survey. In *Approximation Algorithms for NP-Hard Problems*, p. 46–93. Springer.

Cohen, M. C., Keller, P. W., Mirrokni, V., & Zadimoghaddam, M. (2019). Overcommitment in cloud services: Bin packing with chance constraints. *Management Science*, *65*(7), 3255–3271.

Csirik, J., Johnson, D. S., Kenyon, C., Orlin, J. B., Shor, P. W., & Weber, R. R. (2006a). On the sum-of-squares algorithm for bin packing. *Journal of the ACM*, *53*, 1–65.

Csirik, J., Johnson, D. S., Kenyon, C., Orlin, J. B., Shor, P. W., & Weber, R. R. (2006b). On the sum-of-squares algorithm for bin packing. *Journal of the ACM (JACM)*, *53*(1), 1–65.

de la Vega, W. F., & Lueker, G. S. (1981). Bin packing can be solved within 1+epsilon in linear time. *Comb.*, *1*(4), 349–355.

Delorme, M., Iori, M., & Martello, S. (2014). Bin packing and cutting stock problems: mathematical models and exact algorithms. In *Decision models for smarter cities*.

Delorme, M., Iori, M., & Martello, S. (2018). BPPLIB: a library for bin packing and cutting stock problems. *Optim. Lett.*, *12*(2), 235–250.

Diakonikolas, I., Kontonis, V., Tzamos, C., Vakilian, A., & Zarifis, N. (2021). Learning online algorithms with distributional advice. In *ICML*, Vol. 139 of *Proceedings of Machine Learning Research*, pp. 2687–2696. PMLR.

Dósa, G. (2007). The tight bound of first fit decreasing bin-packing algorithm is $ffd(i) \leq 11/9opt(i) + 6/9$. In *Combinatorics, Algorithms, Probabilistic and Experimental Methodologies: First International Symposium, ESCAPE 2007*, pp. 1–11. Springer.

Fukunaga, A. S., & Korf, R. E. (2007). Bin completion algorithms for multicontainer packing, knapsack, and covering problems. *Journal of Artificial Intelligence Research (JAIR)*, *28*, 393–429.

Garey, M. R., & Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

Gent, I. P. (1998). Heuristic solution of open bin packing problems. *Journal of Heuristics*, *3*(4), 299–304.

Gollapudi, S., & Panigrahi, D. (2019). Online algorithms for rent-or-buy with expert advice. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pp. 2319–2327.

Gomes, H. M., Barddal, J. P., Enembreck, F., & Bifet, A. (2017). A survey on ensemble learning for data stream classification. *ACM Computing Surveys (CSUR)*, *50*(2), 1–36.

Gschwind, T., & Irnich, S. (2016). Dual inequalities for stabilized column generation revisited. *INFORMS Journal on Computing*, *28*(1), 175–194.

Gupta, V., & Radovanovic, A. (2020). Interior-point-based online stochastic bin packing. *Operations Research*, *68*(5), 1474–1492.

Hoberg, R., & Rothvoß, T. (2017). A logarithmic additive integrality gap for bin packing. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 2616–2625. SIAM.

Huang, E., & Korf, R. E. (2013). Optimal rectangle packing: An absolute placement approach. *Journal of Artificial Intelligence Research (JAIR)*, *46*, 47–87.

Im, S., Kumar, R., Qaem, M. M., & Purohit, M. (2021). Online knapsack with frequency predictions. In *Proceedings of the 34th Annual Conference on Neural Information Processing Systems 2021 (NuerIPS)*, pp. 2733–2743.

Intel (2010). Implementing and expanding a virtualized environment (white paper)..

Johnson, D. S., Demers, A., Ullman, J. D., Garey, M. R., & Graham, R. L. (1974). Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM Journal on Computing (SICOMP)*, *3*, 256–278.

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

Kamali, S., & López-Ortiz, A. (2015b). All-around near-optimal solutions for the online bin packing problem. In *International Symposium on Algorithms and Computation (ISAAC)*, pp. 727–739.

Kaplan, H., Naori, D., & Raz, D. (2022). Online weighted matching with a sample. In *SODA*, pp. 1247–1272. SIAM.

Korf, R. E. (2002). A new algorithm for optimal bin packing. In *Proceedings of the 18th AAAI Conference on Artificial Intelligence*, pp. 731–736.

Korf, R. E. (2003). An improved algorithm for optimal bin packing. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1252–1258.

Lattanzi, S., Lavastida, T., Moseley, B., & Vassilvitskii, S. (2020). Online scheduling via learned weights. In *Proceedings of the 14th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1859–1877.

Lavastida, T., Moseley, B., Ravi, R., & Xu, C. (2020). Learnable and instance-robust predictions for online matching, flows and load balancing. *CoRR*, *abs/2011.11743*.

Lee, C. C., & Lee, D.-T. (1985). A simple on-line bin-packing algorithm. *Journal of the ACM (JACM)*, *32*(3), 562–572.

Lykouris, T., & Vassilvitskii, S. (2021). Competitive caching with machine learned advice. *Journal of the ACM (JACM)*, *68*(4), 1–25.

Mahdian, M., Nazerzadeh, H., & Saberi, A. (2007). Allocating online advertisement space with unreliable estimates. In *Proceedings of the 8th ACM Conference on Electronic Commerce (EC)*, pp. 288–294. ACM.

Mahdian, M., Nazerzadeh, H., & Saberi, A. (2012). Online optimization with uncertain information. *ACM Trans. Algorithms*, *8*(1), 2:1–2:29.

Mann, Z. A. (2015). Allocation of virtual machines in cloud data centers - A survey of problem models and optimization algorithms. *ACM Comput. Surv.*, *48*(1), 11:1–11:34.

Martello, S., & Toth, P. (1990). Lower bounds and reduction procedures for the bin packing problem. *Discrete Applied Mathematics*, *28*(1), 59–70.

Mikkelsen, J. W. (2016). Randomization can be as helpful as a glimpse of the future in online computation. In *Proceedings of the 43rd International Colloquium on Automata, Languages, and Programming (ICALP)*, Vol. 55, pp. 39:1–39:14.

Mitzenmacher, M., & Vassilvitskii, S. (2020). Algorithms with predictions. In Roughgarden, T. (Ed.), *Beyond the Worst-Case Analysis of Algorithms*, pp. 646–662. Cambridge University Press.

Purohit, M., Svitkina, Z., & Kumar, R. (2018). Improving online algorithms via ML predictions. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS)*, Vol. 31, pp. 9661–9670.

Rhee, W. T., & Talagrand, M. (1993). On line bin packing with items of random size. *Math. Oper. Res.*, *18*(2), 438–445.

Rohatgi, D. (2020). Near-optimal bounds for online caching with machine learned advice. In *Proceedings of the 14th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1834–1845.

Rothvoß, T. (2013). Approximating bin packing within $o(\log opt \cdot \log \log opt)$ bins. In *Proceedings of the 54th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 20–29.

Schoenfield, J. E. (2002). Fast, exact solution of open bin packing problems without linear programming. *Draft, US Army Space and Missile Defense Command*.

Schreiber, E. L., & Korf, R. E. (2013). Improved bin completion for optimal bin packing and number partitioning. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 651–658.

Schwerin, P., & Wäscher, G. (1997). The bin-packing problem: A problem generator and some numerical experiments with FFD packing and MTP. *International Transactions in Operational Research*, *5*(4), 377–389.

Song, W., Xiao, Z., Chen, Q., & Luo, H. (2013). Adaptive resource provisioning for the cloud using online bin packing. *IEEE Transactions on Computers*, *63*(11), 2647–2660.

VMware (2021). Server consolidation.. https://www.vmware.com/ca/solutions/consolidation.html, accessed: 2024-04-17.

Wang, M., Meng, X., & Zhang, L. (2011). Consolidating virtual machines with dynamic bandwidth demand in data centers. In *Proceedings of the 30th IEEE Conference on Computer Communications (INFOCOM)*, pp. 71–75.

Wäscher, G., & Gau, T. (1996). Heuristics for the integer one-dimensional cutting stock problem: A computational study. *Operations-Research-Spektrum*, *18*(3), 131–144.

Wei, A., & Zhang, F. (2020). Optimal robustness-consistency trade-offs for learning-augmented online algorithms. In *Proceedings of the 34th Annual Conference on Neural Information Processing Systems (NeurIPS)*.

Zhao, H., She, Q., Zhu, C., Yang, Y., & Xu, K. (2021). Online 3d bin packing with constrained deep reinforcement learning. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, pp. 741–749.