Online Bin Packing with Predictions

Spyros Angelopoulos[1,2] , **Shahin Kamali**[3] and **Kimia Shadkami**[3]
[1]Centre National de la Recherche Scientifique (CNRS)
[2]LIP6, Sorbonne Université, Paris, France
[3]University of Manitoba, Winnipeg, Manitoba, Canada
spyros.angelopoulos@lip6.fr, shahin.kamali@umanitoba.ca, shadkamk@myumanitoba.ca

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 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. 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 *et al.*, 1996], and has many applications in supply chain management such as capacity planning. Efficient algorithms for the problem have been proposed within AI [Korf, 2002; Schreiber and Korf, 2013].

In this work, we focus on the *online* variant of bin packing, in which the set of items is 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 resource management such as virtual machine placement for server consolidation [Song *et al.*, 2013].

We rely on the standard framework of *competitive analysis* for evaluating the performance of the online algorithms. In fact, as stated in [Coffman *et al.*, 1996], bin packing has served as an early proving ground for this type of analysis. The *competitive ratio* of an online algorithm is the worst-case ratio of the algorithm's cost (total number of opened bins used by the algorithm) 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 unbounded (i.e., the sequence is arbitrarily long) [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 can be inherently erroneous, namely there is a *prediction error* η associated with it. The objective is to design algorithms whose competitive ratio degrades gently as the error increases. Following [Lykouris and Vassilvitskii, 2018], we refer to the competitive ratio of an algorithm with error-free prediction as the *consistency* of the algorithm, and to the competitive ratio with adversarial prediction as its *robustness*. Several online problems have been studied in this setting, including caching [Lykouris and Vassilvitskii, 2018; Rohatgi, 2020], non-clairvoyant scheduling [Purohit *et al.*, 2018; Wei and Zhang, 2020], contract scheduling [Angelopoulos and Kamali, 2021], rent-or-buy problems [Banerjee, 2020; Anand *et al.*, 2020; Gollapudi and Panigrahi, 2019], and time-series search [Angelopoulos *et al.*, 2022]. See also the survey [Mitzenmacher and 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 either assumed ideal and error-free predictions from a very powerful oracle, or only considered tradeoffs between consistency and robustness (see 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-related works on this problem, 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 to items with fractional sizes.

We first present and analyze PROFILEPACKING, which is an algorithm of optimal consistency, but also efficient for small prediction error. As the error grows, this algorithm 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 algorithms which offers a more balanced 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 *et al.*, 2018], but also on distributions studied in the context of offline bin packing, such as the *Weibull* distribution [Castiñeiras *et al.*, 2012]. The results show that our algorithms outperform the known, and efficient algorithms without any predictions that are typically used in practice.

In terms of techniques, we rely on the concept of a *profile set*, which serves as an approximation of the items that are expected to appear in the sequence, given the prediction. This is a natural concept which, perhaps surprisingly, has not been exploited in over 50 years of previous theoretical and experimental work on this problem, and which may be further applicable in other online packing problems, such as multi-dimensional packing [Christensen *et al.*, 2017] and vector packing [Azar *et al.*, 2013]. In fact, our online algorithms can also serve as fast approximations to the offline problem. In terms of the theoretical analysis, it is worth pointing out that it 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 offline optimal solution). Last, in our experiments, the prediction error is a natural byproduct of the learning phase, unlike certain other works in which the prediction error is generated ad-hoc, and in which the error is applied to some perfect prediction obtained by a powerful oracle.

Due to space limitations, we omit certain proofs. All omitted details can be found in [Angelopoulos *et al.*, 2021].

1.2 Related Work

Online bin packing has a long history of study. Simple algorithms such as FIRSTFIT (which places an item into the first bin of sufficient space, and opens a new bin if required), and BESTFIT (which works similarly, except that it places the item into the bin of minimum available capacity which can still fit the item) are 1.7-competitive [Coffman *et al.*, 1996]. The currently best upper bound on the competitive ratio is 1.57829 [Balogh *et al.*, 2018], whereas the best known lower bound is 1.54278 [Balogh *et al.*, 2021]. The results above apply to the standard model with no prediction on the input. Other studies include sequential [György *et al.*, 2010] and stochastic settings [Gupta and Radovanovic, 2020].

The problem has also been studied under the *advice* model [Boyar *et al.*, 2016; Mikkelsen, 2016], in which the online algorithm has access to some error-free information, and the objective is to quantify the tradeoffs between the competitive ratio and the size of this advice (in terms of bits). We emphasize that such studies are only of theoretical interest, not only because the advice is assumed to have no errors, but also because it may encode *any* information, with no learnability considerations (i.e., it may be provided by an omnipotent oracle that knows the optimal solution).

Online bin packing was recently studied under an extension of the advice complexity model, in which the advice may be *untrusted* [Angelopoulos *et al.*, 2020]. Here, the algorithm's performance is evaluated 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. In particular, this model is not concerned with the performance of the algorithm on 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 of advice. In particular, even with error-free predictions, the algorithm of [Angelopoulos *et al.*, 2020] has competitive ratio as large as 1.5.

[Im *et al.*, 2021] studied a related problem under frequency predictions, namely the *online knapsack* problem. In this problem, each item has a value and a size, and the predictions defined in [Im *et al.*, 2021] provide, for each value, an estimate of the total size of all items of that value. For the bin packing problem (in which there is no value), our predictions are precisely the size frequencies. Furthermore, such predictions are PAC-learnable, and in our experiments they are learned by sampling a prefix of the input. Last, in this work, we define a clear notion of error as the L_1 distance between the actual and the predicted frequencies, which helps us quantify the impact of the error on the performance. In contrast, [Im *et al.*, 2021] assumes that the algorithm knows some upper and lower bounds on the predicted frequencies.

2 Online Bin Packing: Model and Predictions

We begin with some preliminary discussions related to our problem. 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 and Korf, 2013; Fukunaga and Korf, 2007; Csirik *et al.*, 2006].

Given an online algorithm A (with no predictions), we denote by $A(\sigma)$ its output (packing) on input σ (i.e., the set of bins opened by A), and by $|A(\sigma)|$ the number of bins in its output. We denote by $\text{OPT}(\sigma)$ the offline optimal algorithm with knowledge of the input sequence. The (asymptotic) competitive ratio of A is defined [Coffman *et al.*, 1996] as $\lim_{n\to\infty} \sup_{\sigma:|\sigma|=n} |A(\sigma)|/|\text{OPT}(\sigma)|$.

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 $f_{x,\sigma} \in [0, 1]$. Our algorithms will use these frequencies as predictions, i.e., 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, $f'_{x,\sigma} \neq f_{x,\sigma}$. 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, 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 $\boldsymbol{f'_\sigma}$. It should be emphasized that unlike the ideal predictions in previous work [Angelopoulos *et al.*, 2020], the item frequencies are PAC-learnable. Namely, 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 (see, e.g., [Canonne, 2020].)

In general, the error η may be bounded by a value H, i.e., $\eta \leq H$. We can thus make a distinction between *H-aware* and *H-oblivious* algorithms, depending on whether the algorithm knows H. Such an upper bound may be estimated e.g., from available data on typical sequences. Unless otherwise specified, we will assume that the algorithm is H-oblivious.

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

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 the optimal packing in constant time [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 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. ProfilePacking packs these special items separately from others, using FirstFit.

3.1 Analysis of ProfilePacking

We first show that in the ideal setting of error-free prediction, ProfilePacking is near-optimal. We will use this result in the analysis of the realistic setting of erroneous predictions. We denote by ϵ any fixed constant less than 0.5. We define m to be any constant such that $m \geq 3\tau_k k/\epsilon$.

Lemma 1. *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. Let $\epsilon' = \epsilon/3$ and note that $m \geq \tau_k k/\epsilon'$. 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 (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 $\text{Opt}(\sigma)$, 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 $\text{Opt}(\sigma)$, along with their corresponding items. For every bin type in $\text{Opt}(\sigma)$, 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 $|\text{Opt}(\sigma)|$. We have that $|N| \leq |\text{Opt}(\sigma)| + (g-1)\tau_k < |\text{Opt}(\sigma)| + n\tau_k/m \leq |\text{Opt}(\sigma)|(1 + \tau_k k/m)$ (since $|\text{Opt}(\sigma)| \geq \lceil n/k \rceil$). We conclude that $|N| \leq (1+\epsilon')|\text{Opt}(\sigma)|$.

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 $\overline{N}$. Let $q = |\overline{N}|$ 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.

Proposition 1. $\lceil n_x/g \rceil \geq \lceil n_x m/n \rceil - 1$.

Proposition 1 implies that for $x \in [1, k]$, $\overline{N}$ packs each item 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 $q + k \geq p$. Note that $q \geq |\textsc{Opt}(\sigma)|/g \geq n/(kg) \geq n/(k\lceil n/m \rceil) > (\lceil n/m \rceil m - m)/(k\lceil n/m \rceil) \geq m/k - m^2/kn > m/k - \epsilon' \geq \tau_k/\epsilon' - \epsilon' > (\tau_k - 1)/\epsilon' > k/\epsilon'$. We thus showed that $q > k/\epsilon'$, and the inequality $p \leq q + k$ implies that $p < q(1 + \epsilon')$. 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 $|PP(\sigma, \boldsymbol{f'})| \leq g \cdot p < g(1 + \epsilon')q \leq (1 + \epsilon')(1 + \epsilon')|\textsc{Opt}(\sigma)| < (1 + 3\epsilon')|\textsc{Opt}(\sigma)| = (1 + \epsilon)|\textsc{Opt}(\sigma)|$. □

We will now use Lemma 1 to prove a more general result.

Theorem 1. *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. Of course, $\boldsymbol{f}$ 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 1 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 $\sum_{x=1}^{k} \mu_x \leq k + \sum_{x=1}^{k} |(f_x - f'_x)m| \leq k + \eta m$. 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 technical proposition.

Proposition 2. $|PP(\sigma, \boldsymbol{f'})| \leq (1 + (2 + 5\epsilon)\eta k + \epsilon)|\textsc{Opt}(\sigma)|$.

□

In addition, we prove the following impossibility result which shows that the dependency in k, as stated in Theorem 1, is unavoidable.

Theorem 2. *Fix any constant $c < 1$. Then for any $\alpha > 0$, with $\alpha \leq 1/k$, any algorithm with frequency predictions that is $(1 + \alpha)$-consistent 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.

4 A Broader Class of Algorithms

In this section we obtain a more general class of algorithms which offer better robustness in comparison to ProfilePacking, at the expense of slightly worse consistency. Let A denote any algorithm of competitive ratio c_A, in the standard online model in which there is no prediction. We will define a class of algorithms based on a parameter $\lambda \in [0, 1]$ which we denote by Hybrid(λ). 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 $m \geq 5\tau_k \max\{k, a + b\}/\epsilon$.

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: count(x), which is the number of items of size x that have been served so far, and 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 ppcount(x) $\leq \lambda \cdot$ count(x), then it is served using ProfilePacking and is declared PP-item. If ppcount(x) $> \lambda \cdot$ count(x), then it is served using A and declared A-item.

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*.

Theorem 3. *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 A.*

One can choose A as the algorithm of the best known competitive ratio [Balogh *et al.*, 2018]. However, algorithms such as the one of [Balogh *et al.*, 2018] belong in a class that is tailored to worst-case competitive analysis and do not tend to perform well in typical instances [Kamali and López-Ortiz, 2015]. For this reason, simple algorithms such as FirstFit and BestFit are preferred in practice [Coffman *et al.*, 1996]. We obtain the following corollary.

Corollary 1. *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*($\lambda$) using* 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 on the error H is known. 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 3 with the observation that as long as $\eta < (c_A - 1 - \epsilon)/(k(2+5\epsilon))$, PROFILEPACKING has a competitive ratio better than c_A.

Corollary 2. *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

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,].

Let r denote the consolidation ratio (but note that this quantity is an integer). We can express the competitive ratio of HYBRID(λ) in Theorem 3, as well as the impossibility result in Theorem 2, so that the term k is replaced with r. We can thus exploit the fact that typically r is much smaller than k, and improve the theoretical analysis of our algorithms.

5.2 Handling Items with Fractional Sizes

As stated in Section 2, items have integral sizes in $[1, k]$. While this is a natural model for many AI applications, we can still handle *fractional* item sizes in $[1, k]$, by treating them as "special" items which 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, we need a measure of "deviation" of the input sequence σ (that may contain fractional items) from a sequence of integral sizes. The most natural approach is to use 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 4. *Let $\lfloor x \rceil$ denote the integer closest to x, and define $d(\sigma) = \sum_{x\in\sigma} |x - \lfloor x \rceil|$. No online algorithm can have 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 $\epsilon > 0$.*

A different measure of "deviation" is 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.

Theorem 5. *Define $\hat{d}(\sigma) = \frac{\sum_{x\in\sigma, x\neq\lfloor x\rceil} x}{\sum_{x\in\sigma} x}$. 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.*

6 Experimental Evaluation

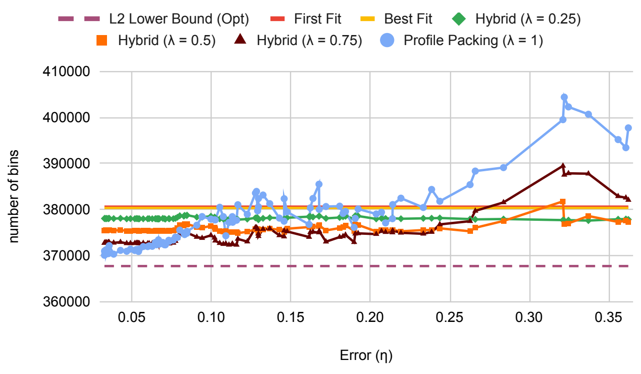
6.1 Benchmarks and Input Generation

Several benchmarks have been used in previous work on (offline) bin packing (see [Castiñeiras *et al.*, 2012] for a list of related work). Many of these 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 [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 and López-Ortiz, 2015].

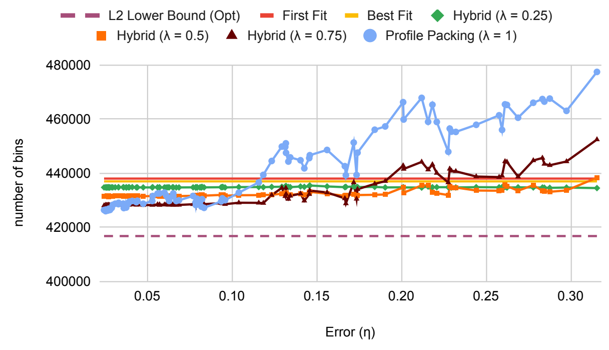
We evaluate our algorithms on two types of benchmarks. The first type is based on the *Weibull* distribution, which was first studied in [Castiñeiras *et al.*, 2012] as a model of several real-world applications of bin packing, e.g., the 2012 ROADEF/EURO Challenge on a data center problem provided by 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 discussed later; we thus set $sc = 1000$, as in [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. Due to space limitations, we report results on the most recent benchmark of the BPPLIB Library, namely the GI Benchmark [Gschwind and Irnich, 2016], and we refer to [Angelopoulos *et al.*, 2021] for additional benchmarks.

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 VM placement (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, for the shape parameter set to $sh = 3.0$. For BPPLIB benchmarks, each item is chosen uniformly and independently at random from the item sizes in one of the benchmark files; this file is also chosen uniformly at random.



(a) Weibull distribution.



(b) GI benchmark from BPPLIB.

Figure 1: Number of bins opened by the algorithms as function of the prediction error.

6.2 Compared Algorithms, Predictions and Error

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

We generate the frequency predictions as follows: For a parameter $b \in \mathbb{N}^+$, we define $\boldsymbol{f'}$ as $\boldsymbol{f}_{\boldsymbol{\sigma}[1,b]}$. In words, we use a prefix of size b of σ so as to estimate the frequencies of item sizes in σ. In our experiments, we consider b of the form $b = \lfloor 100 \cdot 1.05^i \rfloor$, with $i \in [25, 125]$, i.e., 100 prefix sizes. We define the prediction error η as the L_1 distance between the predicted and the actual frequencies. Since we consider 100 distinct values for b, there are 100 possible error values.

As explained earlier, FIRSTFIT and BESTFIT perform very well in practice, and we use them as benchmarks. As often in offline bin packing, we also report the *L2 lower bound* [Martello and Toth, 1990] as a lower-bound estimation of the optimal *offline* bin packing solution (no online or even offline algorithm performs as well as this lower bound).

6.3 Results and Discussion

Figures 1a and 1b depict the cost of the algorithms, as function of the prediction error (For the GI benchmark, the chosen file is file "csBA125_9"). 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.

For both benchmarks, we observe that PROFILEPACKING ($\lambda = 1$) degrades quickly as the error 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 1. 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, HYBRID(λ) dominates FIRSTFIT and BESTFIT for $\lambda \in \{0.25, 0.5\}$, and for practically all values of 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.

7 Conclusion

We gave the first results for online bin packing in a setting in which the algorithm has access to learnable predictions about item frequencies. We believe that our approach can be applicable to generalizations of the problem such as online *3D-packing* [Zhao *et al.*, 2021], which has many applications in transportation logistics, and *vector bin packing* [Azar *et al.*, 2013] which is another important problem in cloud computing. Here, it will be crucial to devise time-efficient profile packing algorithms, since the profile size increases exponentially in the vector dimension.

Acknowledgements

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) [funding reference number DGECR-2018-00059], the grant ANR-19-CE48-0016 from the French National Research Agency (ANR), and the FMJH Program PGMO with support from EDF-THALES-ORANGE.

References

[Anand *et al.*, 2020] K. Anand, R. Ge, and D. Panigrahi. Customizing ML predictions for online algorithms. In *ICML*, pages 303–313, 2020.

[Angelopoulos and Kamali, 2021] Spyros Angelopoulos and Shahin Kamali. Contract scheduling with predictions. In *AAAI*, pages 11726–11733, 2021.

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

[Angelopoulos *et al.*, 2021] Spyros Angelopoulos, Shahin Kamali, and Kimia Shadkami. Online bin packing with predictions. *arXiv 2102.03311*, 2021.

[Angelopoulos *et al.*, 2022] Spyros Angelopoulos, Shahin Kamali, and Dehou Zhang. Online search with best-price and query-based predictions. In *AAAI*, 2022.

[Azar *et al.*, 2013] Y. Azar, I. R. Cohen, S. Kamara, and B. Shepherd. Tight bounds for online vector bin packing. In *STOC*, pages 961–970, 2013.

[Balogh *et al.*, 2018] J. Balogh, J. Békési, G. Dósa, L. Epstein, and A. Levin. A new and improved algorithm for online bin packing. In *ESA*, pages 5:1–5:14, 2018.

[Balogh *et al.*, 2021] János Balogh, József Békési, György Dósa, Leah Epstein, and Asaf Levin. A new lower bound for classic online bin packing. *Algorithmica*, 83(7):2047–2062, 2021.

[Banerjee, 2020] S. Banerjee. Improving online rent-or-buy algorithms with sequential decision making and ML predictions. In *NeurIPS*, 2020.

[Boyar *et al.*, 2016] J. Boyar, S. Kamali, K. S. Larsen, and A. López-Ortiz. Online bin packing with advice. *Algorithmica*, 74(1):507–527, 2016.

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

[Castiñeiras *et al.*, 2012] I. Castiñeiras, M. De Cauwer, and B. O'Sullivan. Weibull-based benchmarks for bin packing. In *CP*, volume 7514, pages 207–222, 2012.

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

[Coffman *et al.*, 1996] E. G. Coffman, M. R. Garey, and D. S. Johnson. Approximation algorithms for bin packing: A survey. In *Approximation Algorithms for NP-Hard Problems*, page 46–93. 1996.

[Csirik *et al.*, 2006] J. Csirik, D. S. Johnson, C. Kenyon, J. B. Orlin, P. W Shor, and R. R. Weber. On the sum-of-squares algorithm for bin packing. *JACM*, 53:1–65, 2006.

[Delorme *et al.*, 2018] M. Delorme, M. Iori, and S. Martello. BPPLIB–a bin packing problem library. http://or.dei.unibo.it/library/bpplib, Accessed: 2022-04-29, 2018.

[Fukunaga and Korf, 2007] A. Fukunaga and R. E. Korf. Bin completion algorithms for multicontainer packing, knapsack, and covering problems. *JAIR*, 28:393–429, 2007.

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

[Gollapudi and Panigrahi, 2019] S. Gollapudi and D. Panigrahi. Online algorithms for rent-or-buy with expert advice. In *ICML*, pages 2319–2327, 2019.

[Gschwind and Irnich, 2016] T. Gschwind and S. Irnich. Dual inequalities for stabilized column generation revisited. *INFORMS*, 28(1):175–194, 2016.

[Gupta and Radovanovic, 2020] V. Gupta and A. Radovanovic. Interior-point-based online stochastic bin packing. *OR*, 68(5):1474–1492, 2020.

[György *et al.*, 2010] A. György, G. Lugosi, and G. Ottucsák. Online sequential bin packing. *JMLR*, 11:89–109, 2010.

[Im *et al.*, 2021] S. Im, R. Kumar, M. M. Qaem, and M. Purohit. Online knapsack with frequency predictions. In *NeurIPS*, 2021.

[Kamali and López-Ortiz, 2015] S. Kamali and A. López-Ortiz. All-around near-optimal solutions for the online bin packing problem. In *ISAAC*, pages 727–739, 2015.

[Korf, 2002] R. E. Korf. A new algorithm for optimal bin packing. In *AAAI*, pages 731–736, 2002.

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

[Mann, 2015] Z. A. Mann. 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, 2015.

[Martello and Toth, 1990] S. Martello and P. Toth. Lower bounds and reduction procedures for the bin packing problem. *DAM*, 28(1):59–70, 1990.

[Mikkelsen, 2016] J. W. Mikkelsen. Randomization can be as helpful as a glimpse of the future in online computation. In *ICALP*, volume 55, pages 39:1–39:14, 2016.

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

[Purohit *et al.*, 2018] M. Purohit, Z. Svitkina, and R. Kumar. Improving online algorithms via ML predictions. In *NeurIPS*, volume 31, pages 9661–9670, 2018.

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

[Schreiber and Korf, 2013] E. L. Schreiber and R. E. Korf. Improved bin completion for optimal bin packing and number partitioning. In *IJCAI*, pages 651–658, 2013.

[Song *et al.*, 2013] W. Song, Z. Xiao, Q. Chen, and H. Luo. Adaptive resource provisioning for the cloud using online bin packing. *TC*, 63(11):2647–2660, 2013.

[VMware,] VMware. Server consolidation (www.vmware.com/ca/solutions/consolidation.html).

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

[Zhao *et al.*, 2021] H. Zhao, Q. She, C. Zhu, Y. Yang, and K. Xu. Online 3d bin packing with constrained deep reinforcement learning. In *AAAI*, pages 741–749, 2021.