Online Algorithms with Advice for the Dual Bin Packing Problem

Marc P. Renault

Abstract This paper studies the problem of maximizing the number of items packed into n bins, known as the dual bin packing problem, in the advice per request model. In general, no online algorithm has a constant competitive ratio for this problem. An online algorithm with 1 bit of advice per request is shown to be 3/2-competitive. Next, for $0 < \varepsilon < 1/2$, an online algorithm with advice that is $(1/(1-\varepsilon))$-competitive and uses $O(1/\varepsilon)$ bits of advice per request is presented.

Keywords online algorithms · online computation with advice · competitive analysis · dual bin packing · multiple knapsack problem

1 Introduction

Online algorithms receive their input in a piecewise manner and make an irrevocable decision on the output with each piece of the input before the next piece is revealed, resulting in a cost or profit to the algorithm. The goal is to either minimize the cost or maximize the profit. Traditionally, online algorithms are studied using *competitive analysis* (cf. [7]). This is the worst case ratio, called the *competitive ratio*, between the cost/profit associated with the output of the algorithm and the cost/profit associated with the optimal output. In the general context of competitive analysis, as compared to the general context of the offline approximation ratio, no computational limits are placed on the online algorithm. The emphasis of competitive analysis is to measure the importance of future knowledge as opposed to the importance of computational power. In particular, competitive analysis quantifies the difference between

Research supported in part by ANR project NeTOC (ANR-11-BS02-0015).

Marc P. Renault
CNRS and Université Paris Diderot, France.
E-mail: mrenault@liafa.univ-paris-diderot.fr

an algorithm with no knowledge of the future and an algorithm with full knowledge of the request sequence.

In many cases, competitive analysis can be seen as too restrictive [7,9,13] as there are many times when it is reasonable to assume that an algorithm has partial knowledge of the future. In order to handle the restrictive nature of competitive analysis, online algorithms have been augmented using ad hoc approaches such as lookahead (e.g. [18]), probability distributions about future requests (e.g. [16,23]), and multiple solutions (e.g. [21,2]). Recently, in part as a generalization of these ad hoc approaches, two models of online computation with advice have been proposed [14,6]. In the model of advice of [6], the online algorithm has access to an advice tape that it can read as needed, where the bits of the advice tape are a function of the input. In the model of [14], a fixed amount of information about the future is given to the algorithm with each request. This is the model of online computation with advice that is used in this paper. In the model of [14], the amount of advice used by an algorithm is quantified by the number of bits needed to encode the advice per request and is called the *bits of advice*. Online computation with advice provides a general model for situations of partial knowledge of the future and has been used to study a variety of classic online problems, e.g. paging [6], the k-server problem [14,24], metrical task system [14], buffer management problems [12, 1], the knapsack problem [5], and the bin packing problem [8,25,3].

This paper studies the dual bin packing problem. The goal of this problem is to maximize the number of items from a sequence (offline algorithms know the whole sequence and can process the items in any order) packed into a fixed number of bins. It is a special case of the multiple knapsack problem (MKP), where the profit is the same for all the items and the capacity is the same for all the knapsacks. The dual bin packing problem, MKP and the Generalized Assignment Problem (GAP) (a generalization of MKP) have been greatly studied in the offline setting (cf. [22]). While these problems have also been studied in the online setting (e.g. [9,4,15,11]), they have been much less studied than in the offline setting given that, in complete generality, no constant competitive algorithms exist for the dual bin packing problem [9,11].

For the maximization problem studied here, an algorithm ALG has a *competitive ratio* of c if, for all finite request sequences σ, $\text{ALG}(\sigma) \geq \frac{1}{c} \cdot \text{OPT}(\sigma) - \alpha$, where $\text{ALG}(\sigma)$ (resp. $\text{OPT}(\sigma)$) is the profit for an online algorithm (resp. offline optimum) to process σ, and α is a constant that does not depend on σ. As defined, the competitive ratio will be greater than or equal to 1. If the additive constant $\alpha = 0$, then the competitive ratio is *strict*. For offline algorithms, the *approximation ratio* can be defined in an analogous manner to that of the competitive ratio.

Related Work without Advice. In [10], the authors show that an offline algorithm, First Fit Increasing(FFI), has an approximation ratio of 4/3. FFI first sorts the input in increasing order and then packs it in a first fit manner. Also, in the offline setting, the problem is NP-complete [17], and a polynomial-time approximation scheme (PTAS) for a generalization of the dual bin packing problem (MKP with equal capacity knapsacks) is presented in [20].

In the general setting for this problem, no online algorithm can pack a constant fraction of the items packed by the offline optimal algorithm regardless of whether the algorithm is deterministic [9] or randomized [11]. Hence, the online work has focused on various settings that either strengthen the algorithm or weaken the adversary, e.g. [9,4,15,11]. When restricted to accommodating sequences (sequences where all the items are packed in an optimal packing), First Fit has an asymptotically tight competitive ratio of 8/5 [4]. Also in [4], the authors consider a variant called Unfair First Fit that has an asymptotically tight competitive ratio of 3/2. The algorithm Unfair First Fit rejects items of size larger than 1/2 (even if there is space to pack it) so long as the ratio between the total number of requests so far and the number of packed items remains above 3/2.

Related Work with Advice. The knapsack problem with advice was studied in [5], using the tape advice model of [6]. Note that, in the per request model of [14] (the model considered here), this problem is trivially solvable, using only 1 bit of advice per request. In [5], for the case of unit profit, Böckenhauer et al. show that 1 bit of advice in total is sufficient for a competitive ratio of 2, and that $\log m$ bits of advice in total are required for a competitive ratio below 2, where m is the length of the request sequence. In the general case, Böckenhauer et al. show that no algorithm is competitive with less than $\log m$ bits of advice. Also, an algorithm is presented that is $(1+\varepsilon)$-competitive with $O((\log m)/\varepsilon)$ bits of advice in total. This result depends on the ability to encode the profit and the weight of the items efficiently in the advice, whereas the $(1+\varepsilon)$-competitive algorithm presented in this paper for the dual bin packing problem (multiple knapsacks) does not have this restriction.

In [25], the authors consider the bin packing problem and scheduling problems. For those problems, the authors present $(1+\varepsilon)$-competitive algorithms that are inspired by (asymptotic) PTAS' for the offline problem, using $O\left(\frac{1}{\varepsilon}\log\left(\frac{1}{\varepsilon}\right)\right)$ bits of advice per request. The $1/(1-\varepsilon)$-competitive algorithm presented here is inspired by the techniques used in [25] and the techniques used for the PTAS presented in [20].

In Boyar et al. [8], they study the bin packing problem with advice, using the model of [6]. They present a 3/2-competitive algorithm, using $\log m$ bits of advice in total and a $(4/3+\varepsilon)$-competitive algorithm, using $2m+o(m)$ bits of advice in total, where m is the length of the request sequence. As both algorithms rely on reading $O(\log(m))$ bits of advice prior to receiving any requests, they would use $O(\log(m))$ bits of advice per request in the online advice model. In addition, they show that an online algorithm with advice requires at least $(m-2n)\log n$ bits of advice to be optimal, where n is the optimal number of bins.

In [3], Angelopoulos et al. show that, for the bin packing problem, a constant amount of advice in total is sufficient for a competitive ratio arbitrarily close to 1.47012. They also show that, in total, linear advice is required for a competitive ratio better than 7/6 (in the tape advice model; in the per request advice model, an algorithm has at least 1 bit of advice per request, i.e. at least linear advice in total).

Results. In this paper, two algorithms with advice for the dual bin packing problem are presented. The first algorithm uses 1 bit of advice per request to indicate which items are to be packed using First Fit. It has a competitive ratio of at most 3/2.

The second algorithm, for $0 < \varepsilon < 1/2$, achieves a competitive ratio of $1/(1-\varepsilon)$, and uses $O\left(\frac{1}{\varepsilon}\right)$ bits of advice per request. It is inspired by the offline PTAS' for GAP problems and the algorithms in [25]. This is of interest as not all online problems allow algorithms with advice that can achieve competitive ratios arbitrarily close to 1 with only a constant amount of advice per request. That is, there are known non-constant bounds on the amount of advice required to be 1-competitive for some problems, e.g. Metrical Task System [14].

The approach of rounding and grouping items used here for the $(1/(1-\varepsilon))$-competitive algorithm is similar to the approach of the $(1+\varepsilon)$-competitive bin packing algorithm in [25]. However, in the algorithm for the dual bin packing problem, bins containing more than $1/\varepsilon$ items are handled differently than bins with less items. This allows for a small savings in the advice used in that we do not have to worry about small items. The main savings comes from the fact that, for the bins with at most $1/\varepsilon$ items, we only need to split them into $1/\varepsilon$ groups as opposed to the $1/\varepsilon^2$ groups for bin packing. This allows for a savings in the amount of advice used per request by a factor of $\log\left(\frac{1}{\varepsilon}\right)$ (up to constant factors). As with the $(1+\varepsilon)$-competitive bin packing algorithm in [25], the advice is based on an optimal packing and could be replaced with a packing produced by a PTAS for the dual bin packing problem. That is, the offline oracle could approximate an optimal solution which is then used to generate the advice and, based on the advice, the offline approximation can be approximated online.

Lastly, in [8], the authors show that at least $(m-2n)\log n$ bits of advice in total (at least $\left(1-\frac{2n}{m}\right)\log n$ bits per request), where m is the length of the request sequence and n is the optimal number of bins, are needed for any online bin packing algorithm with advice to be optimal. Note that by using the same techniques, the same bound on the amount of advice needed to be optimal can be shown for the dual bin packing problem. That is, the lower bound construction for the dual bin packing problem uses the same sets of items as in the lower bound construction for the bin packing problem, where the optimal bin packing solution uses n bins. Hence, the optimal packing for the dual bin packing problem would pack all the items in the n bins whereas an algorithm using less than $(m-2n)\log n$ bits of advice in total will reject at least 1 item. This follows from a similar argument to the claim in the bin packing lower bound that shows that a bin packing algorithm using less than $(m-2n)\log n$ bits of advice in total will require at least $n+1$ bins.

2 Preliminaries

The *dual bin packing problem* consists of a sequence σ of items and a set B of n bins (initially empty and numbered from 1 to n) with capacity 1. Each $r_j \in \sigma$ is an item with size $0 < s(r_j) \leq 1$ and $|\sigma| = m$. The goal is to maximize the number of the items of σ assigned to the n bins such that, for all bins $b_i \in B$, $\sum_{r_j \in b_i} s(r_j) \leq 1$. That is, to pack as many items as possible in the n bins. Items that are not packed are called *rejected items*.

An assignment of some items of σ to the set of bins B is called a *packing*. Unassigned items of σ are the rejected items. A bin is *valid* if the total size of the packed items in the bin is at most 1. An item r_i is *packed* in a bin b if r_i is assigned to the bin $b \in B$ in a packing. A bin b can *accommodate* an item r_i if the bin remains valid if r_i is assigned to b. In this case, r_i *fits* into b. A packing is valid if all the bins in B are valid. An item $r_i \in b$, where b is a bin in B, may be denoted by $r_i \in B$.

In the algorithms presented here, a common heuristics for bin packing, FIRST FIT (FF) (cf. [19]), is used.

Definition 1 (FIRST FIT) To pack an item r_i in a FIRST FIT manner, r_i is packed in the first bin that can accommodate r_i in an ordering of the bins. If no such bin exists, r_i is rejected.

For an algorithm ALG and a request sequence σ, the set of rejected items is denoted by R_σ^{ALG}, and the set of accepted items is denoted by A_σ^{ALG}. Let $p_\sigma^{\text{ALG}} = \lfloor s(r_j)^{-1} \rfloor$, where $r_j \in R_\sigma^{\text{ALG}}$ is the smallest item rejected by ALG. Let P_σ^{ALG} be a packing produced by ALG for the items of σ. For a given P_σ^{ALG}, let $k_{\sigma,i}^{\text{ALG}}$ be the number of items in the i-th bin, let $B_{\sigma,i}^{\text{ALG}}$ be the set of items packed in the i-th bin, and let $v_{\sigma,i}^{\text{ALG}}$ be the total size of the items packed in $B_{\sigma,i}^{\text{ALG}}$. For two algorithms, ALG$_1$ and ALG$_2$, such that $R_\sigma^{\text{ALG}_2} \subseteq R_\sigma^{\text{ALG}_1}$, the set $D_\sigma^{\text{ALG}_1,\text{ALG}_2}$ is defined to be $R_\sigma^{\text{ALG}_1} \setminus R_\sigma^{\text{ALG}_2}$. When it is clear by the context, the superscripts and subscripts will be suppressed.

As in [14], a *deterministic online algorithm with advice* is an online algorithm that is augmented with a query function to a finite advice space U, where $|U| = 2^b$ and $b \geq 0$ is the amount of advice per request. The advice received for request i is defined by $u_i : R^* \to U$, where R^* is the infinite set of all finite request sequences. A sequence of pairs (g_i, u_i) defines the algorithm, where $g_i : R^i \times U^i \to A_i$ is the action function and A_i is the set of available actions. For a request $r_i \in \sigma = r_1, r_2, \ldots$, the action a_i of a online deterministic algorithm with b bits of advice is $g_i(r_1, \ldots, r_i, u_1(\sigma), \ldots, u_i(\sigma))$, i.e. a function of all the requests and the advice received so far.

Throughout the paper, log is assumed to be base 2.

3 A 1.5-Competitive Online Algorithm with 1 bit of Advice per Request

In this section, an online algorithm with advice called SUBSEQUENCE FIRST FIT (SFF) that uses 1 bit of advice is presented. First, an offline algorithm called Restricted Subsequence First Fit (rsff) is shown to be 3/2-competitive. This bound is shown to hold for sff as, by definition, sff will pack the same number of items as rsff.

Restricted Subsequence First Fit. Initially, define $\sigma^{\text{RSFF}} = \sigma$. Then, simulate the ff packing of σ^{RSFF}. If $|R^{\text{FF}}(\sigma^{\text{RSFF}})| > 0$, remove the largest item from σ^{RSFF} (for ties, remove the item with the smallest index). Repeat the process using ff until all the items of σ^{RSFF} can be packed in the n bins. This is the packing produced by rsff.

The key properties of rsff are that the items are packed using ff and that the largest items have been rejected, i.e. the size of every rejected item is at least as large as any packed item. The following lemma shows that, on average, the number of items in every bin for a packing produced by rsff is at least $p^{\text{RSFF}} = \lfloor s(r_j)^{-1} \rfloor$, where r_j is the smallest rejected item.

Lemma 1 *For any σ, $\sum_{i=1}^{n} k_i^{\text{RSFF}} \geq p^{\text{RSFF}} n$.*

Proof Let σ^{RSFF} be the subsequence of σ that is packed by rsff and let r_j be the smallest item rejected by rsff. By the definition of rsff, $|R^{\text{FF}}_{\sigma^{\text{RSFF}}}| = 0$ and $|R^{\text{FF}}_{\sigma^{\text{RSFF}} \cup \{r_j\}}| > 0$. Therefore, $|P^{\text{FF}}_{\sigma^{\text{RSFF}}}| \geq |P^{\text{FF}}_{\sigma^{\text{RSFF}} \cup \{r_j\}}|$ and, to prove the lemma, it suffices to show that $k_i^{\text{FF}} \geq p^{\text{RSFF}}$ for all $i \in [n]$ of the ff packing of $\sigma^{\text{RSFF}} \cup \{r_j\}$.

Consider the ff packing of $\sigma^{\text{RSFF}} \cup \{r_j\}$. By the definition of $\sigma^{\text{RSFF}} \cup \{r_j\}$, no item is larger than $s(r_j)$. Suppose, for the sake of contradiction, that $k_i^{\text{FF}} < p := p^{\text{RSFF}}$ for some $1 \leq i \leq n$. The packed items in the i-th bin have size less than or equal to $s(r_j)$. The free space in the i-th bin is greater than or equal to $1 - (p-1)(s(r_j)) \geq s(r_j)$ implying that an item packed in the bins B_{i+1} to B_n or a rejected item should have been packed in the i-th bin by ff, which is a contradiction. □

The following lemma shows that there always exists an optimal packing that rejects the largest items.

Lemma 2 *For any σ, there exists an optimal packing such that R^{OPT} is the set of the $|R^{\text{OPT}}|$ largest items (for ties, the item with the smallest index is rejected).*

Proof Fix an optimal packing for σ. Let $r_j \in R^{\text{OPT}}$ be the smallest rejected item. Consider the largest $r_i \in A^{\text{OPT}}$. If $s(r_i) > s(r_j)$, replace r_i by r_j in the packing and add r_i to R^{OPT}. If $s(r_i) = s(r_j)$ and $i < j$, replace r_i by r_j in the packing and add r_i to R^{OPT}. Repeat this procedure until, for all $r_i \in A^{\text{OPT}}$, $s(r_j) > s(r_i)$, or $s(r_j) > s(r_i)$ and $j < i$. The packing resulting from this procedure is still optimal and all the rejected items are at least as large as all the packed items. □

In the following, it is always assumed that opt uses the optimal packing as described by Lemma 2. In the next lemma, this optimal packing is considered, and it is shown that the difference between the number of rejected items for rsff and an optimal algorithm is less than n.

Lemma 3 *For any σ, there exists an* OPT *such that* $|D| = |R^{\text{RSFF}} \setminus R^{\text{OPT}}| \leq n-1$.

Proof Fix an optimal packing for σ. By Lemma 2 and the definition of RSFF, we can assume that $R^{\text{OPT}} \subseteq R^{\text{RSFF}}$ and R^{OPT} is the set of the $|R^{\text{OPT}}|$ largest items of R^{RSFF} (ties are broken by rejecting the item with the smallest index). Let $s(r_j)$ be the size of the smallest rejected item in R^{RSFF}. The total size of the items in D is at most the empty space of P^{RSFF} which is less than $s(r_j)n$. Therefore, $s(r_j)(|R^{\text{RSFF}}|-|R^{\text{OPT}}|) \leq \sum_{\eta \in R^{\text{RSFF}} \setminus R^{\text{OPT}}} s(\eta) < s(r_j)n$. Hence, $|R^{\text{RSFF}} \setminus R^{\text{OPT}}| = |R^{\text{RSFF}}| - |R^{\text{OPT}}| < n$. □

We are now ready to prove the upper bound on RSFF.

Theorem 1 *For any σ, the approximation ratio of* RSFF *is at most* $3/2$.

Proof If there are empty bins, then RSFF has packed all the items and $\frac{\text{OPT}(\sigma)}{\text{RSFF}(\sigma)} = 1$. In the following, we only consider instances with rejected items. Also, we always assume that OPT uses the optimal packing as described by Lemma 2.

Let $s(r_j)$ be the size of the smallest rejected item in R^{RSFF} and let $p := p^{\text{RSFF}}$.

Case 1: $p \geq 2$ ($s(r_j) \leq \frac{1}{2}$). In the packing by RSFF, all the bins contain at least p items on average by Lemma 1. Therefore, $|A^{\text{RSFF}}| \geq pn$. Thus, using Lemma 3, the approximation ratio is

$$\frac{\text{OPT}(\sigma)}{\text{RSFF}(\sigma)} = \frac{|A^{\text{RSFF}}| + |D|}{|A^{\text{RSFF}}|} < \frac{pn+n}{pn} = \frac{p+1}{p} \leq \frac{3}{2}\text{ , for } p \geq 2.$$

Case 2: $p = 1$ ($\frac{1}{2} < s(r_j)$). Let q be the number of bins packed with at least 2 items and let A^L be the set of accepted large items (size $> 1/2$). The first fit packing guarantees that at most one of the $n-q$ bins with a single item contains a small item (size $\leq 1/2$). Therefore, $|A^L| \geq n-q-1$.

By the definition of OPT (Lemma 2), OPT packs the items of D and A^L (all of which are large). Since at most one large item can be packed per bin and the number of bins is n, we have that $|D| + |A^L| \leq n$. Adding this to $|D| \leq n-1$ from Lemma 3, we have that

$$|D| \leq \frac{2n-1-|A^L|}{2} \leq \frac{n+q}{2}\ . \tag{1}$$

Using (1) and the fact that $|A^{\text{RSFF}}| \geq 2q + (n-q) = n+q$, we get that the approximation ratio is

$$\frac{\text{OPT}(\sigma)}{\text{RSFF}(\sigma)} = \frac{|A^{\text{RSFF}}| + |D|}{|A^{\text{RSFF}}|} \leq \frac{n+q+\frac{n+q}{2}}{n+q} = \frac{3}{2}\ .$$

□

Now, an online algorithm with 1 bit of advice per request is defined and shown to always pack exactly the same number items as RSFF.

Subsequence First Fit *with 1 bit of Advice.* If the bit of advice is 1, pack the requested item using ff. Otherwise, the bit is 0 and the item is rejected. The advice is based on the final subsequence σ^{RSFF} of σ that would be packed by rsff as defined previously. The advice bit for r_i is defined to be 1 if $r_i \in \sigma^{\text{RSFF}}$ and 0 otherwise.

Theorem 2 *For any σ, the competitive ratio of* sff *is at most* $3/2$*.*

Proof The final sequence of items of σ^{RSFF} is defined such that they can be packed, using ff, into the n bins. This fact and Theorem 2 imply that sff is $3/2$-competitive. □

In the following theorem, we show that the presented algorithm has a competitive ratio of at least $4/3$.

Theorem 3 sff *has a competitive ratio of at least* $4/3$*.*

Proof For an ε, $0 < \varepsilon < 1/6$, and an an even number of bins n, let σ be n requests of size $1/2 - \varepsilon$ followed by n requests of size $1/2 + \varepsilon$. An optimal packing includes all the items for a total of $|\sigma| := m = 2n$ items. sff packs the first n requests in $n/2$ bins and then packs $n/2$ of the remaining items for a competitive ratio of at least $\frac{2n}{3n/2} = 4/3$. □

4 A $(1/(1-\varepsilon))$-Competitive Algorithm with Advice

In this section, an algorithm with advice, denoted by dbpa (Dual Bin Packing Advice), is presented that has a competitive ratio of $1/(1-2\varepsilon)$, for $0 < \varepsilon < 1/2$, and uses $O\left(\frac{1}{\varepsilon}\right)$ bits of advice per request. This shows that an algorithm can arbitrarily approach a competitive ratio of 1 with advice bits that are inversely proportional to the approximation guarantee. It should be noted that if the amount of bits of advice per request is $\log(n+1)$, then the trivial algorithm with advice can be used to be optimal. That is, the trivial algorithm is given a bin number ($n+1$ for a rejected item), as advice that indicates the bin into which the item should be packed. Also, the competitive ratio of dbpa is only better than that of sff for $\varepsilon < 1/6$. For ease of presentation, $1/\varepsilon$ is assumed to be a natural number.

Initially, an algorithm adbpa (Asymptotic Dual Bin Packing Advice) is presented that is $(1/(1-\varepsilon))$-competitive and uses $O\left(\frac{1}{\varepsilon}\right)$ bits of advice per request. Then, with a slight modification to adbpa and one additional bit of advice, the algorithm dbpa is presented that has a strict competitive ratio of $(1/(1-2\varepsilon))$.

4.1 A $(1/(1-\varepsilon))$-Competitive Algorithm.

The algorithm presented in this section is called adbpa and has a competitive ratio of $1/(1-\varepsilon)$. It uses $O\left(\frac{1}{\varepsilon}\right)$ bits of advice per request for $0 < \varepsilon < 1/2$.

The advice for ADBPA is defined such that ADBPA can distinguish between two types of items in an optimal packing. They are: (1) items that belong to bins with more than $1/\varepsilon$ items and (2) items that belong to bins with at most $1/\varepsilon$ items. Another bit of advice will indicate for each item of type (1) whether to pack it in a first fit manner into a subset of the available bins or reject it. The items of type (2) will be classified into groups based on the size of the item (each group being an ε fraction of the total number of items). The advice will indicate for each item of type (2) the index of the group to which the item belongs and the type of bin (see bin pattern below) in which it should be packed. The only items of type (2) that are rejected are the items from the group containing the largest items. The algorithm and the advice are formally defined in the following.

Let P^{OPT} be an optimal packing for σ. We assume without loss of generality that OPT is the optimal algorithm described in Lemma 2. We split the bins of P^{OPT} into two sets of bins, P_1 and P_2, such that all the bins of P_1 have more than $1/\varepsilon$ items packed in each bin and $P_2 = P^{\text{OPT}} \setminus P_1$.

Let $\sigma^{>\frac{1}{\varepsilon}}$ be the subsequence of items from σ that are packed in the bins of P_1. Further, let σ^{SFF} be the subsequence of items from $\sigma^{>\frac{1}{\varepsilon}}$ that would be packed by SFF (see Section 3) if given $|P_1|$ bins to pack $\sigma^{>\frac{1}{\varepsilon}}$.

Let $\sigma^{\leq\frac{1}{\varepsilon}}$ be the subsequence of items from σ that are packed in the bins of P_2. The items of $\sigma^{\leq\frac{1}{\varepsilon}}$ are sorted in non-increasing order of size and, based on that ordering, the items are partitioned into $\left(|\sigma^{\leq\frac{1}{\varepsilon}}| \mod 1/\varepsilon\right)$ consecutive groups of size $\left\lceil \varepsilon|\sigma^{\leq\frac{1}{\varepsilon}}|\right\rceil$ followed by $\left(1/\varepsilon - |\sigma^{\leq\frac{1}{\varepsilon}}| \mod 1/\varepsilon\right)$ consecutive groups of size $\left\lfloor \varepsilon|\sigma^{\leq\frac{1}{\varepsilon}}|\right\rfloor$. The items are assigned a type from 1 to $1/\varepsilon$ based on their group in the sorted order. As the bins of P_2 have at most $1/\varepsilon$ items, the packing of each bin can be described by a $1/\varepsilon$ length vector, where the entries are the types of the packed items. Note that the item types have values in the range $[1, 1/\varepsilon]$. This vector will be called a *bin pattern*.

Items that are rejected by OPT, i.e. $r_i \notin \sigma^{>\frac{1}{\varepsilon}} \cup \sigma^{\leq\frac{1}{\varepsilon}}$, are also assigned a type of 1. This is a technical detail to ensure that ADBPA rejects all the items rejected by OPT without using an additional bit of advice.

We now define a process that assigns bin patterns to the items. The assigned bin patterns will be received as part of the advice to the algorithm. Let Q be the multiset of the bin patterns of P_2. Each item of type $\tau > 1$ in $\sigma^{\leq\frac{1}{\varepsilon}}$ is assigned a bin pattern in Q as follows. Let B be a set of $|P_2|$ bins to which bin patterns will be assigned. Initially, the bins of B have no patterns assigned. Consider each item $r_i \in \sigma^{\leq\frac{1}{\varepsilon}}$ with type $t_i > 1$ in the order it arrives as defined by σ. Let $t'_i = t_i - 1$ and pack r_i in a bin b in B such that the number of items packed in b with type t_i is less than the number of items of type t'_i as specified by the pattern of b. If no such bin exists, pack r_i in an empty bin b without an assigned pattern. Assign a pattern $p \in Q$ that contains type t'_i to b, and set $Q = Q \setminus p$. In both cases, r_i is assigned the bin pattern of b. Note that the assignment and the packing are feasible since the number of items of type τ

is at least the number of items of type $\tau - 1$, and all the items of type τ are smaller in size than the items of type $\tau - 1$.

We now define the online algorithm with advice. ADBPA maintains two sets of bins L_1 and L_2. As ADBPA processes σ, the n bins will be assigned to either L_1 or L_2 such that, after processing σ, $|L_1| = |P_1|$ and $|L_2| = |P_2|$. Initially, none of the bins are assigned to either set. L_1 is the set of bins that contains the items in $\sigma^{\mathrm{SFF}} \subseteq \sigma^{>\frac{1}{\varepsilon}}$, and L_2 is the set of bins that contains the type $\tau > 1$ items in $\sigma^{\leq\frac{1}{\varepsilon}}$. As bins are added to L_2, they will be assigned bin patterns. For each r_i in σ, ADBPA gets a bit of advice per request to indicate if $r_i \in \sigma^{>\frac{1}{\varepsilon}}$ or not and packs r_i as follows.

$r_i \in \sigma^{>\frac{1}{\varepsilon}}$: For each $r_i \in \sigma^{>\frac{1}{\varepsilon}}$, an additional bit of advice indicates if $r_i \in \sigma^{\mathrm{SFF}}$. If so, ADBPA packs r_i into the bins of L_1 in a first fit manner. If r_i does not fit, an unassigned bin is assigned to L_1 and r_i is packed in that bin. Otherwise, the additional bit indicates that $r_i \notin \sigma^{\mathrm{SFF}}$ and r_i is rejected.

$r_i \notin \sigma^{>\frac{1}{\varepsilon}}$: For each $r_i \notin \sigma^{>\frac{1}{\varepsilon}}$, ADBPA receives the item type, t_i, as advice. If $t_i = 1$, the item is rejected. Otherwise, let $t'_i = t_i - 1$. In this case, ADBPA also receives an assigned bin pattern, p_i (as defined previously) as advice, where t'_i is an entry of p_i. The algorithm packs r_i in a bin $b_j \in L_2$ with a pattern p_i, containing less items of type t_i than there are entries of t'_i in p_i. If no such bin exists, ADBPA assigns an unassigned bin b_j to L_2, assigns b_j the pattern p_i, and packs r_i in b_j.

From the definition of ADBPA, we have that the items of σ^{SFF} are packed in $|P_1|$ bins, and the items of $\sigma^{\leq\frac{1}{\varepsilon}}$ are packed in $|P_2|$ which implies the following fact.

Fact 1 ADBPA *will use n bins to pack the non-rejected items of σ.*

In the next two lemmas, we bound from below the number of items of $\sigma^{>\frac{1}{\varepsilon}}$ and the number of items of $\sigma^{\leq\frac{1}{\varepsilon}}$ that ADBPA packs as compared to OPT. For $x \in \{\sigma^{>\frac{1}{\varepsilon}}, \sigma^{\leq\frac{1}{\varepsilon}}\}$, let ADBPA^x and OPT^x be the number of items packed from σ^x by ADBPA and OPT, respectively. Note that, $\mathrm{ADBPA}(\sigma) = \mathrm{ADBPA}^{\sigma^{>\frac{1}{\varepsilon}}} + \mathrm{ADBPA}^{\sigma^{\leq\frac{1}{\varepsilon}}}$ and $\mathrm{OPT}(\sigma) = \mathrm{OPT}^{\sigma^{>\frac{1}{\varepsilon}}} + \mathrm{OPT}^{\sigma^{\leq\frac{1}{\varepsilon}}} = |\sigma^{>\frac{1}{\varepsilon}}| + |\sigma^{\leq\frac{1}{\varepsilon}}|$.

Lemma 4 $\mathrm{ADBPA}^{\sigma^{>\frac{1}{\varepsilon}}} > (1-\varepsilon)\mathrm{OPT}^{\sigma^{>\frac{1}{\varepsilon}}}$.

Proof By the definition of $\sigma^{>\frac{1}{\varepsilon}}$, OPT packs all the items of $\sigma^{>\frac{1}{\varepsilon}}$ in $|P_1|$ bins. Therefore, for any optimal packing of $\sigma^{>\frac{1}{\varepsilon}}$ in $|P_1|$ bins, the set of rejected items is empty, i.e. $\left|R^{\mathrm{OPT}}_{\sigma^{>\frac{1}{\varepsilon}}}\right| = 0$. For $|P_1|$ bins and request sequence $\sigma^{>\frac{1}{\varepsilon}}$, Lemma 3 gives

$$\left|R^{\mathrm{RSFF}}_{\sigma^{>\frac{1}{\varepsilon}}} \setminus R^{\mathrm{OPT}}_{\sigma^{>\frac{1}{\varepsilon}}}\right| = \left|R^{\mathrm{RSFF}}_{\sigma^{>\frac{1}{\varepsilon}}}\right| < |P_1| \ . \tag{2}$$

Let $R^{\sigma^{>\frac{1}{\varepsilon}}}$ be the rejected items of $\sigma^{>\frac{1}{\varepsilon}}$ by ADBPA when run on σ. ADBPA packs the items of $\sigma^{>\frac{1}{\varepsilon}}$ into $|P_1|$ bins. This produces the same packing as SFF and, by the definition of SFF, the same packing as RSFF when packing $\sigma^{>\frac{1}{\varepsilon}}$

into $|P_1|$ bins. This implies that $\left|R^{\text{RSFF}}_{\sigma^{>\frac{1}{\varepsilon}}}\right| = |R^{\sigma^{>\frac{1}{\varepsilon}}}|$ when the number of bins used by RSFF to pack $\sigma^{>\frac{1}{\varepsilon}}$ is $|P_1|$. Further, using (2), we have that

$$|R^{\sigma^{>\frac{1}{\varepsilon}}}| < |P_1| < \varepsilon \text{OPT}^{\sigma^{>\frac{1}{\varepsilon}}} ,$$

where the last inequality follows from the fact that $\text{OPT}^{\sigma^{>\frac{1}{\varepsilon}}} > \frac{1}{\varepsilon}|P_1|$ since, in OPT, the items of $\sigma^{>\frac{1}{\varepsilon}}$ are packed into $|P_1|$ bins such that there are more than $1/\varepsilon$ items in each bin. Therefore,

$$\text{ADBPA}^{\sigma^{>\frac{1}{\varepsilon}}} = \text{OPT}^{\sigma^{>\frac{1}{\varepsilon}}} - |R^{\sigma^{>\frac{1}{\varepsilon}}}| > (1-\varepsilon)\text{OPT}^{\sigma^{>\frac{1}{\varepsilon}}}$$

□

Lemma 5 $\text{ADBPA}^{\sigma^{\leq\frac{1}{\varepsilon}}} > (1-\varepsilon)\text{OPT}^{\sigma^{\leq\frac{1}{\varepsilon}}} - 1$.

Proof ADBPA, by definition, will reject the type 1 items of $\sigma^{\leq\frac{1}{\varepsilon}}$ and pack the rest of the items of $\sigma^{\leq\frac{1}{\varepsilon}}$. Therefore, there are $\left\lceil \varepsilon|\sigma^{\leq\frac{1}{\varepsilon}}|\right\rceil < \varepsilon\text{OPT}^{\sigma^{\leq\frac{1}{\varepsilon}}} + 1$ items rejected by ADBPA. So, $\text{ADBPA}^{\sigma^{\leq\frac{1}{\varepsilon}}} > (1-\varepsilon)\text{OPT}^{\sigma^{\leq\frac{1}{\varepsilon}}} - 1$. □

Formal Advice Definition. The bin pattern vectors, padded to a length of exactly $1/\varepsilon$, have at most $\frac{1}{\varepsilon}+1$ different possible values per entry ($\frac{1}{\varepsilon}$ values for each item type and an additional value for the padding). Since the order of the items in a bin does not matter, vectors will have the entries ordered from largest to smallest. The number of patterns is less than the number of ways to pull $1/\varepsilon$ names out of a hat with $1/\varepsilon+1$ names, where repetitions are allowed and order is not significant. Therefore, there are at most $\binom{2/\varepsilon}{1/\varepsilon} \leq \left(\frac{2e/\varepsilon}{1/\varepsilon}\right)^{1/\varepsilon} = (2e)^{\frac{1}{\varepsilon}}$ different bin patterns and at most $\left\lceil \frac{\log(2e)}{\varepsilon} \right\rceil$ bits of advice are needed to specify a pattern from an enumeration of all possible patterns, where e is Euler's number.

Per request, the advice string will be xyz, where x is 1 bit in length, y has a length of $\lceil \log(1/\varepsilon) \rceil$ bits to indicate the item type, and z has a length of $\left\lceil \frac{\log(2e)}{\varepsilon} \right\rceil$ bits. The advice string xyz is defined for request r_i as follows.

$$x = \begin{cases} 1, & \text{if } r_i \in \sigma^{>\frac{1}{\varepsilon}} \ . \\ 0, & \text{otherwise.} \end{cases}$$

$$y = \begin{cases} 1, & \text{if } x = 1 \text{ and } r_i \in \sigma^{\text{SFF}} \ . \\ 0, & \text{if } x = 1 \text{ and } r_i \notin \sigma^{\text{SFF}} \ . \\ \text{The type of } r_i \text{ encoded in binary,} & \text{if } x = 0 \ . \end{cases}$$

$$z = \begin{cases} 0, & \text{if } x = 1 \text{ or } z = 1 \ . \\ \text{The index of the bin pattern assigned to } r_i \text{ encoded in binary,} & \text{otherwise.} \end{cases}$$

Note that when $x = 1$, the last bit of y is the second advice bit used to pack the $\sigma^{>\frac{1}{\varepsilon}}$ items. Immediate from the advice definition is the following fact.

Fact 2 ADBPA *uses $O(\frac{1}{\varepsilon})$ bits of advice per request.*

Finally, we show that ADBPA has a competitive ratio of $1/(1-\varepsilon)$.

Lemma 6 *For any ε, $0 < \varepsilon < 1/2$,* ADBPA$(\sigma) > (1-\varepsilon)OPT(\sigma) - 1$.

Proof Using Lemma 4 and Lemma 5, we get that

$$\begin{aligned}\text{ADBPA}(\sigma) &= \text{ADBPA}^{\sigma^{>\frac{1}{\varepsilon}}} + \text{ADBPA}^{\sigma^{\leq\frac{1}{\varepsilon}}} \\ &> (1-\varepsilon)\text{OPT}^{\sigma^{>\frac{1}{\varepsilon}}} + (1-\varepsilon)\text{OPT}^{\sigma^{\leq\frac{1}{\varepsilon}}} - 1 \\ &= (1-\varepsilon)\text{OPT}(\sigma) - 1\ .\end{aligned}$$

□

Fact 2 and Lemma 6 immediately imply the following theorem.

Theorem 4 *For any ε, $0 < \varepsilon < 1/2$,* ADBPA *has a competitive ratio of $\frac{1}{1-\varepsilon}$ and uses $O(\frac{1}{\varepsilon})$ bits of advice per request.*

4.2 A strict $(1/(1-2\varepsilon))$-Competitive Algorithm.

The algorithm DBPA is defined in this section. It behaves in two different manners, depending on the optimal number of packed items. One bit, denoted by w, is used to distinguish between the two cases and is sent as advice with each item. In the first case, DBPA runs ADBPA as previously described. In the second case, DBPA is able to pack the items optimally in one of two different ways. The choice of which way to pack the items depends on ε and n. The details of these two cases are as follows.

Case 1: OPT$(\sigma) > 1/\varepsilon$ $(w = 0)$. DBPA will run ADBPA as described previously. The only difference is that the advice per request for ADBPA is prepended with an additional bit for w. Since OPT$(\sigma) > 1/\varepsilon$, we get the following corollary to Lemma 6 for this case.

Corollary 1 *If $w = 0$, then, for any ε, $0 < \varepsilon < 1/2$,* DBPA$(\sigma) > (1 - 2\varepsilon)OPT(\sigma)$.

Case 2: OPT$(\sigma) \leq 1/\varepsilon$ $(w = 1)$. For this case, the algorithm can determine if the number of bins is more than $1/\varepsilon$ as ε and the number of bins is known. If the number of bins given to the algorithm is at least $1/\varepsilon$, then the algorithm packs one item per bin and is optimal.

Otherwise, the number of bins is less then $1/\varepsilon$. For each $r_i \in \sigma$, we define the advice (after w) to be the bin number in which r_i is packed in an optimal packing. This can be done with $\lceil\log(1/\varepsilon)\rceil$ bits. DBPA will pack r_i into the bin as specified by the advice. Note that the packing produced in this case is optimal.

Immediate from the definition of the algorithm and the advice, and Corollary 1, we get the following Theorem.

Theorem 5 *For any* ε*,* $0 < \varepsilon < 1/2$*,* DBPA *has a strict competitive ratio of* $\frac{1}{1-2\varepsilon}$ *and uses* $O\left(\frac{1}{\varepsilon}\right)$ *bits of advice per request.*

5 Conclusion

In this paper, we considered the dual bin packing problem and showed that an algorithm that uses 1 bit of advice per request has a competitive ratio of 3/2. This algorithm ensures that the largest items are rejected and packs the remaining items in a reasonably efficient manner (FIRST FIT). It would be interesting to close the gap for this algorithm and, in particular, to show which properties (e.g. FF and rejecting the largest) are necessary for an algorithm to be 4/3-competitive.

We showed that it is possible to approach a competitive ratio of 1 with a constant amount of bits of advice per request, yet $\Omega(\log n)$ bits of advice per request are required to be optimal. A natural follow up to this work would be to explore more general versions of this problem. Many of the offline versions of the generalized problems have PTAS'. It would be interesting to see if the techniques here could be extended to the more generalized versions of the problems.

Finally, it would be interesting to exploring this problem in the advice complexity model of [6]. In this model, the algorithm has access to a tape of advice bits which can be read as needed thus allowing for sub-linear (in the size of input) advice in total.

Acknowledgements

I would like to thank Reza Dorrigiv for suggesting this as an interesting problem to study. Also, I would like to thank Reza Dorrigiv and Norbert Zeh for useful preliminary discussions and Adi Rosén for helpful comments.

References

1. Adamaszek, A., Renault, M.P., Rosén, A., van Stee, R.: Reordering buffer management with advice. In: Approximation and Online Algorithms - 11th International Workshop, WAOA 2013, Sophia Antipolis, France, September 5-6, 2013, Revised Selected Papers, pp. 132–143 (2013)
2. Albers, S., Hellwig, M.: Online makespan minimization with parallel schedules. In: Algorithm Theory - SWAT 2014 - 14th Scandinavian Symposium and Workshops, Copenhagen, Denmark, July 2-4, 2014. Proceedings, pp. 13–25 (2014)

3. Angelopoulos, S., Dürr, C., Kamali, S., Renault, M.P., Rosén, A.: Online bin packing with advice of small size. In: Algorithms and Data Structures - 14th International Symposium, WADS 2015, Victoria, BC, Canada, August 5-7, 2015. Proceedings, pp. 40–53 (2015)
4. Azar, Y., Boyar, J., Epstein, L., Favrholdt, L.M., Larsen, K.S., Nielsen, M.N.: Fair versus unrestricted bin packing. Algorithmica **34**(2), 181–196 (2002)
5. Böckenhauer, H., Komm, D., Královic, R., Rossmanith, P.: The online knapsack problem: Advice and randomization. Theor. Comput. Sci. **527**, 61–72 (2014)
6. Böckenhauer, H.J., Komm, D., Královic, R., Královic, R., Mömke, T.: On the advice complexity of online problems. In: ISAAC, pp. 331–340 (2009)
7. Borodin, A., El-Yaniv, R.: Online computation and competitive analysis. Cambridge University Press, New York, NY, USA (1998)
8. Boyar, J., Kamali, S., Larsen, K.S., López-Ortiz, A.: Online bin packing with advice. Algorithmica **74**(1), 507–527 (2016)
9. Boyar, J., Larsen, K.S., Nielsen, M.N.: The accommodating function: A generalization of the competitive ratio. SIAM J. Comput. **31**(1), 233–258 (2001)
10. Coffman, E.G., Leung, J.Y.T., Ting, D.W.: Bin packing: Maximizing the number of pieces packed. Acta Inf. **9**, 263–271 (1978)
11. Cygan, M., Jez, L., Sgall, J.: Online knapsack revisited. Theory Comput. Syst. **58**(1), 153–190 (2016)
12. Dorrigiv, R., He, M., Zeh, N.: On the advice complexity of buffer management. In: ISAAC, pp. 136–145 (2012)
13. Dorrigiv, R., López-Ortiz, A.: A survey of performance measures for on-line algorithms. SIGACT News **36**(3), 67–81 (2005)
14. Emek, Y., Fraigniaud, P., Korman, A., Rosén, A.: Online computation with advice. Theor. Comput. Sci. **412**(24), 2642–2656 (2011)
15. Epstein, L., Favrholdt, L.M.: On-line maximizing the number of items packed in variable-sized bins. Acta Cybern. **16**(1), 57–66 (2003)
16. Feldman, J., Mehta, A., Mirrokni, V., Muthukrishnan, S.: Online stochastic matching: Beating 1-1/e. In: Proceedings of the 2009 50th Annual IEEE Symposium on Foundations of Computer Science, FOCS '09, pp. 117–126. IEEE Computer Society, Washington, DC, USA (2009)
17. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman (1979)
18. Grove, E.F.: Online bin packing with lookahead. In: Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms, SODA '95, pp. 430–436. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (1995)
19. Johnson, D.: Near-optimal bin packing algorithms. Ph.D. thesis, MIT (1973)
20. Kellerer, H.: A polynomial time approximation scheme for the multiple knapsack problem. In: RANDOM-APPROX, pp. 51–62 (1999)
21. Kellerer, H., Kotov, V., Speranza, M.G., Tuza, Z.: Semi on-line algorithms for the partition problem. Oper. Res. Lett. **21**(5), 235–242 (1997)
22. Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack problems. Springer (2004)
23. Mahdian, M., Yan, Q.: Online bipartite matching with random arrivals: An approach based on strongly factor-revealing lps. In: Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing, STOC '11, pp. 597–606. ACM, New York, NY, USA (2011)
24. Renault, M.P., Rosén, A.: On online algorithms with advice for the k-server problem. Theory Comput. Syst. **56**(1), 3–21 (2015)
25. Renault, M.P., Rosén, A., van Stee, R.: Online algorithms with advice for bin packing and scheduling problems. Theor. Comput. Sci. **600**, 155–170 (2015)