Theory of Computing Systems manuscript No.
(will be inserted by the editor)

Online Knapsack Revisited

Marek Cygan · **Łukasz Jeż** · **Jiří Sgall**

the date of receipt and acceptance should be inserted later

Keywords online algorithms · competitive analysis · online knapsack · dual bin packing

CR Subject Classification Theory of computation/Design and analysis of algorithms/Online algorithms

Abstract We investigate the online variant of the *(Multiple) Knapsack* Problem: an algorithm is to pack items, of arbitrary sizes and profits, in k knapsacks (bins) without exceeding the capacity of any bin. We study two objective functions: the sum and the maximum of profits over all bins. With either objective, our problem statement captures and generalizes previously studied problems, e.g. *Dual Bin Packing* [1,6] in case of the sum and *Removable Knapsack* [10,11] in case of the maximum. Following previous studies, we consider two variants, depending on whether the algorithm is allowed to remove items (forever) from its bins or not, and two special cases where the profit of an item is a function of its size, in addition to the general setting.

We study both deterministic and randomized algorithms; for the latter, we consider both the oblivious and the adaptive adversary model. We classify each variant as either admitting $O(1)$-competitive algorithms or not. We develop simple $O(1)$-competitive algorithms for some cases of the max-objective variant believed to be intractable because only 1-bin deterministic algorithms were considered before.

A preliminary version of this article, authored by M. Cygan and Ł. Jeż, appeared in the proceedings of *11th Workshop on Approximation and Online Algorithms* (2013)

M. Cygan
Institute of Informatics, University of Warsaw, Poland
E-mail: cygan@mimuw.edu.pl

Ł. Jeż
School of Computer Science, Tel Aviv University, Israel
and
Institute of Computer Science, University of Wrocław, Poland
E-mail: lje@cs.uni.wroc.pl

J. Sgall
Computer Science Institute of Charles University, Faculty of Mathematics and Physics, Praha, Czech Republic
E-mail: sgall@iuuk.mff.cuni.cz

1 Introduction

Knapsack Problems form a fundamental class of problems in computer science, and entire books [16, 13] are dedicated to them. We focus on an online variant of the *Multiple Knapsack* problem, in which k knapsacks (bins) of integer *capacities* C_i and a set of items of arbitrary integer *sizes* and *profits* are given, and the goal is to pack a set of items into the bins, so that their total profit is maximized.

In the online variant that we study, we relax the assumption that all numbers are integers, and allow real numbers instead. On the other hand, we restrict the problem significantly by assuming that *all bins have the same capacity*. This assumption is motivated by the fact that without any restrictions of this kind, we would have to take into account instances where all but one bin have capacities so small that they can accommodate no item. Clearly, such instances are at least as hard as those with only a single bin, for which there are many impossibility results. With all bins having the same capacity, we normalize it and the item sizes so that the capacity of each bin is 1.

The online algorithm is given the items one after another, at which time it learns the item's size and profit. When an item is presented, the algorithm is required to immediately accept or reject it. In the former case, it also has to choose a bin to put the item into. The algorithm is not allowed to move the items between the bins at any time but it is allowed to remove any item from any bin at any time in the *removable* variant. Once removed, an item is treated as if it was never accepted, i.e., it cannot be placed back in any bin. In the *non-removable* variant, an algorithm is not allowed to remove items from bins, i.e. once an item is placed in a bin, it remains there forever.

In both variants, we consider two objective functions: the sum and the maximum profit over all bins, denoted *sum*-objective and *max*-objective respectively; the profit of the bin is the sum of profit of all items currently in it. Note that both objective functions coincide for $k = 1$; in such case, we do not state which one we are considering. Moreover, we remark that under the *max*-objective increasing k cannot increase the competitive ratio, cf. Section 2.1.

The study of online variants of the knapsack problem is motivated by the richness of its applications and the need of efficiently solving very large instances, for which data may only be accessible at bunches from external storage [15]. Thus the potential for improving the competitive ratio at little computational cost by removals, randomization or additional bins motivates their study.

We investigate general instances and, following existing literature, some restricted classes. When each item in the instance has profit equal to its size, we call the instance or case *proportional*; in previous studies such instances have been called either uniform or unweighted [10, 11]. When each item in the instance has the same profit, irrespective of the size, we call the instance or case *unit*.

1.1 Previous and Related Results

Our problem statement is general enough to capture various online knapsack problems studied before. We discuss these as well as some related results below, distinguishing between the two objective functions. We also note that as the problem(s) that we consider are fundamental and our algorithmic solutions simple, similar algorithms have appeared in various contexts, for example in ad auctions [2, 7].

1.1.1 Max-*objective*

No *deterministic* algorithm for 1-bin non-removable problem (the classic online knapsack problem) is $O(1)$-competitive [14,15], even in the proportional and the unit case. For the 1-bin proportional case a tight bound of 2 is known for randomized algorithms in the oblivious adversary model [4]. However, that work focuses on the advice complexity rather than randomized algorithms, and prior studies focused on deterministic algorithms for the removable variant.

Even in the removable variant, there is no $O(1)$-competitive 1-bin deterministic algorithm [10,11], so Iwama et al. [10] focused on deterministic algorithms for its proportional variant. They obtained a tight bound of $\phi \approx 1.618$ for the 1-bin variant, and also a tight bound of (approximately) 1.3815 for the k-bin variant where $k \geq 2$; we note that their algorithm (which achieves the tight bound) for the latter case uses only two bins even if more are allowed. In a recent article, Han et al. [9] studied a more general setting of 1-bin deterministic algorithms for the removable variant under convex function, i.e., where there is a fixed convex function such that, for each item, its profit is the value of this function applied on the item's size. In another recent work, Han et al. [8] studied 1-bin randomized algorithms for the general and proportional case of the removable variant. Independently, they obtained some of the results given in this work (cf. Table 1 in Section 1.2 for details), as well as a 1-bin $10/7$-competitive randomized algorithm for the proportional case.

Iwama et al. [11] also studied the effect of resource augmentation for 1-bin settings, i.e., letting a deterministic algorithm use a bin of capacity $C > 1$ while comparing its profit with the optimum profit attained with a bin of capacity 1. These results are incomparable with ours: on the one hand, a single bin of capacity C is superior to C bins of capacity 1 each, on the other, C need not be an integer; in particular, in all the variants where such a large bin helps, $C \leq 2$ suffices to achieve ratio 1.

Partially fractional online knapsack is a related problem in which the algorithm is allowed to accept any fraction $x \in [0,1]$ of any given item. This x fraction of the item takes only x fraction of the item's size and gives x fraction of its profit. In removable variant, for each (fraction of an) item, the algorithm can remove its entire fraction. The general case of removable variant of this problem has been studied for 1 bin of capacity $C \geq 1$ (i.e., with resource augmentation) [17].

1.1.2 Sum-*objective*

The *sum*-objective has been studied for the *Dual Bin Packing* problem [1,6], which coincides with the unit case of our non-removable variant. Since there is no $O(1)$-competitive algorithm for this setting [6], these studies either focused on the competitive ratio as a function of the largest item size allowed (as a fraction of the bin capacity) [6] or considered the so called *accommodating sequences* [1], in which OPT can fit all the items in its k bins or, in a generalization, $c \cdot k$ bins for some constant c.

1.2 Our Results

We study both objective functions, *sum*-objective and *max*-objective, for both variants, removable and non-removable, for both general and restricted instance classes. Furthermore, we study both deterministic and randomized algorithms, distinguishing between the "standard" oblivious and the less common adaptive adversary model for the latter. The difference

between the two adversaries is that an oblivious adversary has to fix the entire instance in advance without knowing the algorithm's random choices, and clearly can solve such instance optimally. On the other hand, the adaptive (online) adversary is allowed to create the instance in an online manner, i.e., knowing what the algorithm did so far (in particular, knowing the outcome of its random choices), but is also required to solve the instance online. For more details and relations, see the original article that distinguished the models [3] or the textbook on online algorithms [5]. Finally, we consider different numbers of bins. We classify all these settings into tractable, i.e., admitting $O(1)$-competitive algorithms, or intractable, i.e., not admitting $O(1)$-competitive algorithms, by giving upper bounds for all tractable settings and lower bounds for all settings, or pointing to any previously known bounds.

All relevant results are summarized in Table 1 (and its caption; some are not listed in the table). Here we only note a few features. The proportional case, studied by Iwama et al. for one and multiple bins [10] with removals, is tractable even without removals if $k > 1$ or randomization is allowed. The general case of the removable variant exhibits similar properties.

We remark that we obtain simple $O(1)$-competitive randomized algorithms for some 1-bin settings that are intractable for deterministic algorithms. This is a rare phenomenon, exhibited only by a single other *natural* problem we are aware of. Namely, online throughput maximization on a single machine [12], where the constant hidden in $O(1)$ is large and the analysis quite involved.

2 Preliminaries

For an item i, we denote its profit by p_i and its size by s_i. In case of sequences of items, say $i_1, i_2, \ldots, i_n$, we shall abbreviate p_{i_j} and s_{i_j} to p_j and s_j. For any positive integer k, the set of bins available to a k-bin algorithm is $\mathscr{B}_k = \{B_1, B_2, \ldots, B_k\}$; if $k = 1$ we shall denote the sole bin by B rather than B_1. Given a set of items X, we shall denote the total profit and the total size of the items in X by $p(X)$ and $s(X)$, i.e., $p(X) = \sum_{e \in X} p_e$ and $s(X) = \sum_{e \in X} s_e$ respectively. In most of our analyses, the set X will be the set of all items in a bin; in such case we shall simply write $p(B)$ and $s(B)$ for a bin B, ignoring the particular instance and step in notation, since they will be clear from the context. We also say that $s(B)$ is the size of the bin. We extend this notation further to sets of bins: if $\mathscr{B}$ is a subset of the set of all bins, then $p(\mathscr{B}) = \sum_{B \in \mathscr{B}} p(B)$ denotes the total profit of all the items in the bins of $\mathscr{B}$ and similarly $s(\mathscr{B}) = \sum_{B \in \mathscr{B}} s(B)$ denotes their total size. For an algorithm ALG, we will use ALG to denote both the algorithm and its profit; in particular this applies to the offline optimum, denoted by OPT. Unless otherwise stated, all the algorithms are presented assuming there are k bins available.

2.1 Max-objective and different numbers of bins

Note that in the *max*-objective problem, the optimum offline solution can use only a single bin. Therefore, in the oblivious adversary model, an R-competitive k-bin algorithm (which may be randomized) for any variant of the problem remains R-competitive when $\ell > k$ bins are allowed, by ignoring the extra bins. On the other hand, running an arbitrary k_1-bin algorithm and k_2-bin algorithm in parallel does not result in a $(k_1 + k_2)$-bin algorithm, since the former two algorithms might *conflict* by accepting the very same item.

var	obj	case	deterministic lower	upper	randomized (oblivious) lower	upper
R	1	gen	∞	–	$\frac{e+1}{e} \approx 1.37$ (T8) & [8]	2 (S 3.1) & [8]
		prop	$\phi \approx 1.62$	$\phi \approx 1.62$	1.25 (T9) & [8]	$\frac{10}{7} \approx 1.43$ [8]
		unit	1	1	1	1
	max	gen	$\phi \approx 1.62$ (T7)	2 (T2)	1	2
		prop	≈ 1.38 [10]	≈ 1.38 [10]	1	≈ 1.38
		unit	1	1	1	1
	$\sum$	gen	$\frac{7}{6} \approx 1.17$	$3 + O(\frac{1}{k})$ (T1)	$\frac{7}{6} \approx 1.17$	3 (T1)
		prop	$\frac{8}{7} \approx 1.14$	1.6 (T4)	$\frac{8}{7} \approx 1.14$ (T5)	1.6
		unit	$\frac{7}{6} \approx 1.17$	1.5 (T3)	$\frac{7}{6} \approx 1.17$ [1]	1.5
NR	1	gen	∞	–	∞	–
		prop	∞	–	2 [4]	2 [4]
		unit	∞	–	∞	–
	max	gen	∞	–	∞	–
		prop	2	2 [4]	2 (T11)	2
		unit	∞	–	∞ [6]	–
	$\sum$	gen	∞	–	∞	–
		prop	$1 + \ln 2 \approx 1.69$	2 (T10)	$1 + \ln 2$ (T12)	2
		unit	∞	–	∞ [6]	–

Table 1 Summary of previous and our results. Column 'var' describes the variant (R: removable, NR: non-removable), 'obj' the objective (while the meaning of max and $\sum$ is clear, note that it implicitly regards $k > 1$; 1 as 'obj' means the single bin setting, in which both objectives coincide), in the next column 'gen' and 'prop' stand for the general and the proportional case respectively. In 'max' rows, all upper bounds are attained for $k = 2$ whereas all lower bounds hold for all k. For this reason, Theorem 6, which gives a lower bound matching Theorem 2 for $k = 2$ does not appear in the table. In '$\sum$' rows, the upper bounds hold for all $k > 1$, whereas the lower bounds hold for infinitely many k, as well as in the limit. The table does not present our results on the adaptive adversary model (Theorems 13 and 14), which show that for $k = 1$ randomization does not help against such adversary. Bounds without a reference are either folklore or follow from other entries. References in brackets point to sections (S) or theorems (T) in this work, those in square brackets to other articles.

However, if $k < \ell$, one can obtain many k-bin algorithms from a given ℓ-bin algorithm through projections: let ALG be an ℓ-bin algorithm Then, for any $\mathscr{B} \subset \mathscr{B}_\ell$, let $\pi(\text{ALG}, \mathscr{B})$ be a $|\mathscr{B}|$-bin algorithm defined as follows: for every bin $B \in \mathscr{B}$, $\pi(\text{ALG}, \mathscr{B})$ simulates ALG on B, ignoring the items that ALG does not place in B. For the oblivious adversary model, it is easy to see that if ALG is an R-competitive ℓ-bin algorithm, then choosing $\mathscr{B}'$ of cardinality k from its bins uniformly at random yields an $R \cdot \ell/k$-competitive barely random algorithm.

If $k = 1$, using ℓ single-bin algorithms is less restrictive, since they are allowed to conflict. In particular, if ALG_1 and ALG_2 are (deterministic) 1-bin algorithms such that at any point $\text{ALG}_1 + \text{ALG}_2 \geq \text{OPT}$, then a barely random 1-bin algorithm that simulates one of them chosen uniformly at random, is 2-competitive.

3 Removable variants

3.1 Upper bounds

First, we consider general instances for both objective functions in the removable model. We develop a $3k/(k-1)$-competitive deterministic algorithm for the *sum*-objective, whose barely random variant is 3-competitive irrespective of k; see the statement of Theorem 1 for

exact upper bound on the algorithm's competitive ratio. Then we consider the *max*-objective and note that for $k = 2$ bins the same algorithm is 2-competitive (and this ratio is optimal). Moreover, while the aforementioned randomized algorithm is 3-competitive for all k, we note that for $k = 1$ its competitive ratio can be improved to 2 by altering the probability distribution.

Moreover, we give improved results for restricted instances and the *sum*-objective. For the unit case, we give a 1.5-competitive deterministic algorithm, and the proportional case, we give a 1.6-competitive deterministic algorithm. Both classes of instances are completely resolved for the *max*-objective as optimum bounds for the proportional case were given by Iwama et al. [10], whereas the unit case is trivial: a deterministic 1-bin algorithm that tentatively puts all new items in its bin and then repeatedly removes the largest items while the bin overflows is 1-competitive.

3.1.1 General case

To state our algorithm, let us classify items as follows: an item is *small* if its size is at most $1/2$, otherwise it is *large*. Essentially, we are going to use two well-known algorithms, extended to many bins, to handle each type of item.

Algorithm GREEDY: When a new item e is issued, while there is not enough space in any bin to put e, remove the item $e' \in \{e\} \cup \bigcup_{i=1,\dots,k} B_i$ that minimizes $p(e')/s(e')$ from its bin, removing the most-recent item in case of ties. Stop when either e is removed or there is a bin that can accommodate it. In the latter cases, put e into that bin.

Algorithm PGREEDY: Maintain the k most profitable items, one per bin, as follows. When a new item e is issued, put it in an empty bin if there is one. Otherwise, if $p(e) > \min_{i=1,\dots,k} p(B_i)$, replace the minimum-profit item with e.

Recall that, with some arbitrary numbering of the algorithm's bins, B_i denotes its i-th bin and $\mathscr{B}_i$ denotes the set of its first i bins. Moreover, $p(X)$ and $s(X)$ denote the total profit and the total size of items in the set X; we abuse this notation slightly and use it also for bins and sets of bins.

Lemma 1 *Suppose that* GREEDY *is given a sequence of n items. Let $e_1, e_2, \dots, e_n$ denote these items sorted non-increasingly by their profit to size ratio, breaking ties so that older items appear first. Then, after processing the given sequence,* GREEDY *will have $e_1, e_2, \dots, e_{n_0}$ in its bins, where either $n_0 = n$ or $s(B_i) > 1 - \max_{1 \le j \le n} s_j$ for $i = 1, \dots, k$. Finally, for any set of items X from the sequence,*

$$p(X) \le \frac{p(\mathscr{B}_k)}{s(\mathscr{B}_k)} \cdot \max\{s(\mathscr{B}_k),\ s(X)\} \ .$$

Proof The fact that GREEDY keeps a prefix of the sequence in its bins follows from its definition. The bound on $s(B_i)$ is also trivial: if $n_0 < n$, then some items were removed or ignored. Let e_j be the last item that upon release caused a removal of some item, including itself. Then $s(B_i) > 1 - s_j$ held for each i when e_j was issued, and since then no item was removed. The last inequality, i.e., the bound on $p(X)$, is a consequence of the fact that the profit to size ratios of the items in $\mathscr{B}_k$ dominate those of any other set, by the definition of GREEDY. □

Our algorithm is a combination of the two algorithms above.

Algorithm MULTIGREEDY: Use PGREEDY on $\lceil k/3 \rceil$ of the bins for large items and GREEDY on the remaining $\lfloor 2k/3 \rfloor$ bins for small items.

The randomized variant, with probability $1/3$, uses PGREEDY on all the bins for large items, ignoring small items, and, with probability $2/3$, uses GREEDY on all the bins for small items, ignoring large items.

Theorem 1 MULTIGREEDY *is* $R(k)$*-competitive for the* sum*-objective, where*

$$R(k) = \begin{cases} 3 & \text{if } k \equiv 0 \\ 3 + \frac{3}{k-1} & \text{if } k \equiv 1 \\ 3 + \frac{3}{2k-1} & \text{if } k \equiv 2 \end{cases} \pmod 3 \ .$$

A barely random variant of the algorithm is 3*-competitive for all* k*.*

Proof Let L (L^*) and S (S^*) denote ALG's (OPT's) profit for large and small items, respectively.

We focus on the randomized variant first. When packing the large items by PGREEDY, we have $L \geq L^*$ as the algorithm keeps the most profitable items. When packing the small items by GREEDY, if not all of them are packed, every bin is at least half-full and the items of highest profit to size ratio are used, thus $S \geq S^*/2$ (a special case of Lemma 1). The expected profit is $L/3 + 2S/3 \geq L^*/3 + S^*/3$, and the algorithm is 3-competitive.

Now we analyze the deterministic MULTIGREEDY algorithm. Note that the following invariant is clearly maintained: the MULTIGREEDY algorithm has the $\lceil k/3 \rceil$ most profitable large items. Moreover, by Lemma 1, at each step MULTIGREEDY either has all the small items, or all of its $\lfloor 2k/3 \rfloor$ bins dedicated to small items are at least half-full.

To bound the competitive ratio, we observe that $(L^* + S^*)/(L + S) \leq \max(L^*/L, S^*/S)$, hence we focus on large and small items separately. OPT can have at most k large items, so $L^*/L \leq k/\lceil k/3 \rceil \leq 3$. As for the small items, Lemma 1 implies $S^*/S \leq 2k/\lfloor 2k/3 \rfloor$. The bound on MULTIGREEDY's competitive ratio is equal to the bound on S^*/S. □

Theorem 2 MULTIGREEDY *with* 2 *bins is* 2*-competitive for the* max*-objective.*

Proof We refine the proof of Theorem 1. Note that, for $k = 2$, MULTIGREEDY uses one bin for large items and one bin for small items. On the other hand, OPT only uses a single bin. If OPT does not use any large item in its solution, then the 2-competitiveness of MULTIGREEDY follows from the proof of Theorem 1.

Otherwise, OPT uses a single large item, say x. Then MULTIGREEDY has some large item y in its "large" bin, and $p_y \geq p_x$. Furthermore, the total size of OPT's small items is at most $1 - s_x < 1/2$. With this in mind, it follows again from the proof of Theorem 1 that $S \geq S^*$. Hence, $L + S \geq L^* + S^* = \text{OPT}$, and consequently $\text{MULTIGREEDY} = \max\{L, S\} \geq \text{OPT}/2$. □

By Theorem 1, a barely random 1-bin variant of MULTIGREEDY is 3-competitive. But a trivial 2-approximation algorithm for *Knapsack* gives rise to a 2-competitive barely random 1-bin algorithm: toss a fair coin and, depending on the result, simulate either 1-bin GREEDY or 1-bin PGREEDY on all items. This was also independently observed by Han et al. [8].

3.1.2 Unit case

The following algorithm is actually the algorithm GREEDY which we studied for the general case simplified by the unit profit assumption and, furthermore, we specify that, if an item can be packed in more bins, the FIRSTFIT rule is applied.

Algorithm UNITGREEDY: When a new item e is issued, try the following, in this order:

1. Add e to the first bin that can accommodate it without removals.
2. Let m be the item of maximum size in UNITGREEDY's bins; in case of ties let, m be the most recent item. If $s(e) \geq s(m)$, reject e. If $s(e) < s(m)$, replace m by e.

Theorem 3 UNITGREEDY *is* 1.5*-competitive for the* sum*-objective on unit instances.*

Proof We begin by observing that UNITGREEDY maintains in its packing a set of n_0 smallest items for some n_0. To prove this, let r be the smallest item rejected or replaced (if there is any) at any moment; in case of ties, let r be the one that was removed or replaced earliest. We claim that: (1) every item of size strictly smaller than $s(r)$ that was already given or of size exactly $s(r)$ that was given before r is packed by UNITGREEDY, and (2) every item of size strictly greater than $s(r)$ or of size exactly $s(r)$ that was given after r is removed or rejected and the remaining space in each bin of UNITGREEDY is strictly smaller than $s(r)$. This holds when the first item is removed or rejected, since we remove the largest item. When an item i with $s(i) \geq r$ arrives, it does not fit and is rejected, so the invariant continues to hold. If $s(i) < r$, there are two possible cases: either the item i fits without any removal or the largest item among the ones packed and i is removed. In the first case, the invariant holds trivially since $s(i) < s(r)$ by (2). In the second case, let r' be the item that was removed. If $s(r') \geq s(r)$, then again the invariant holds trivially. If $s(r') < s(r)$, then note that it follows from (1) that all items of size strictly smaller than $s(r)$ were packed until now. As r' is the largest (and most recent in case of ties) among those, it follows that until now no items of sizes in $(s(r'),\ s(r))$ arrived, so the invariant holds for r'.

We may assume w.l.o.g. that also OPT packs a maximal set of smallest items. It follows from the claim that each extra item of OPT does not fit into any of UNITGREEDY's bins, and hence OPT can have at most $k-1$ such items due to total volume constraint. (Moreover, OPT can have any extra items only if all UNITGREEDY's bins are nonempty, which implies a ratio of at most $2-1/k$, on which we further improve.)

Recall that an item is small if its size is at most $1/2$ and large otherwise. We observe that UNITGREEDY has at most a single bin containing a single small item and no other items. This follows, since the number of items in any bin never decreases. Hence, if there is a bin with only a single small item, another small item will neither be placed in a new bin nor replace a large item.

Now we prove the theorem. As UNITGREEDY trivially gives an optimum solution if it removes no items, we focus on the case when it does remove some items. Hence, all of its bins are non-empty at the end. We claim that the number of OPT's extra items is no larger than half the number of UNITGREEDY's items. If UNITGREEDY has no large items, this follows since it has at least $2k-1$ items, and there are at most $k-1$ extra items. Suppose that UNITGREEDY has $L > 0$ large items. Then any extra item is also large. Hence, there are at most $k-L$ extra items. As UNITGREEDY $\geq L+2(k-L)-1 = 2k-L-1 \geq 2(k-L)$, the claim follows. □

3.1.3 Proportional case—the simplified asymptotic version

In this and the next sections, we give an improved deterministic algorithm PROFIT for proportional instances under the *sum*-objective (and $k > 1$), our most technically involved upper bound.

Before we describe the algorithm, let us classify the items by their sizes as follows: Let $\alpha \in (1/2, 1)$ be a given parameter and let $\beta = 1 - \alpha$. An item is *small* if its size is in $[0, \beta]$, *medium* if it is in $(\beta, 1/2)$, *big* if it is in $[1/2, \alpha)$, and *huge* if it is in $[\alpha, 1]$. This classification is a refinement of the simplistic one we used previously, and it is used for this result only.

To explain the main ideas of the algorithm, we first sketch a simplified version which only approaches the ratio of 1.6 asymptotically for a large m. We give this simplified version in a parametric form, to see that our later choice of α is optimal for the given approach.

Suppose that we want to achieve the ratio of $1/\alpha$. A natural way to do this is to try to fill each bin to at least α, as the size is equal to the profit and thus the optimum cannot get larger profit than 1 per bin. If all but a constant number of bins have size at least α, the algorithm is asymptotically $1/\alpha$-competitive. (Recall that the size of the bin means the size of all items packed into it, not to be confused with the capacity which is always 1.)

First we give the algorithm and its analysis assuming that no small items arrive as this is the most important case. Each big item is packed in an empty bin if such a bin is available; otherwise the item is rejected. Each huge item is packed into an empty bin; if no empty bin is available, then into a bin with a big item, which is removed; if no such bin exists, then the item is rejected. Medium items are packed in pairs. More precisely, a medium item is packed into a bin with a single medium item if such a bin exists. Otherwise it is packed into an empty bin, or else into a bin with a big item, which is removed, or it is rejected if no such bin exists.

If a huge or medium item is rejected, it means that each bin, with a possible exception of a single bin with one medium item, contains either a huge item or two medium items. Thus the size of each bin except one is at least $\min\{\alpha, 2\beta\}$ and the algorithm is asymptotically $1/\alpha$-competitive, as long as $2\beta \geq \alpha$.

Otherwise all the huge and medium items are packed. If also all the big items are packed, the solution is optimal. It remains to handle the case when some big item is removed or rejected, which implies that no bin is empty. We use a charging argument which gives a matching of the bins in the optimal (offline) solution bins to the bins of the algorithm so that the sizes of the matched bins have the required ratio. Actually, it is a bit more complicated, as we compare not single bins but typically pairs of bins. Any bin in the optimal solution that contains a huge or medium item is charged to a bin of the algorithm containing that item; if the bin in the optimum contains more than one medium item, choose arbitrarily. The remaining bins of the optimum are charged arbitrarily one-to-one to the remaining bins of algorithm. Each bin of the algorithm that is charged two bins of the optimum is paired with one of the uncharged bins; this is well-defined as the number of the bins is the same in the algorithm and in the optimum. In this simplified analysis, we ignore the bin with one medium item, if it exists (there is at most one such bin), and the bin of the optimum that is charged to it. Then each remaining bin has size at least $1/2$. First we analyze the pairs of bins: The two optimal bins have total size at most 2, while the two bins of the algorithm have size at least $2\beta + 1/2$, as one of the bins has two medium items and the other one has at least a big item. Thus the ratio is $1/(\beta + 1/4)$. Any remaining optimal bin with a huge or medium item is charged to a bin of size α, giving the ratio at most $1/\alpha$. Any remaining optimal bin with no huge or medium item contains only a single big item, thus the ratio is at most $\alpha/(1/2) = 2\alpha$. The best overall ratio is achieved when $\alpha = \beta + 1/4 = (1 - \alpha) + 1/4$,

which gives $\alpha = 5/8$ and $1/\alpha = 1.6$; note that the ratio in the remaining case of charging is $2\alpha = 10/8$, which is far from tight, and also $2\beta \geq \alpha$, which we needed in the previous analysis.

It remains to extend the algorithm to the small items, which is a bit technical but not hard. We combine them with the big items as much as possible. We maintain the invariant that each bin contains small items of total size at most β. A new small item is packed into one of the bins with some small item(s) if that does not violate the invariant. If no such bin exists, it is packed into some bin that contains only a big item (it always fits, as $\alpha + \beta \leq 1$). If no such bin exists either, it is packed into an empty bin. If that is not possible, we need to violate the invariant for one bin: We put the small items greedily into bins with only small items so that they are filled one by one to have size of at least α; this is possible, since the small items have size at most $\beta = 1 - \alpha$. The packing of other items is modified as follows: A big item is put into a bin with only small items of total size at most β if such a bin exists; otherwise it is put in an empty bin. Huge and medium items are handled as before, except that if no empty bin remains, they are also put into the bins with only small items of total size at most β, possibly removing some of the small items if necessary. Only if no such bin with small items remains, they are packed into a bin with a single big item and that item is removed. The algorithm stops whenever all but three bins have size at least α; in such a case the asymptotic $1/\alpha$ ratio is guaranteed.

The important property of this packing of small items is that bins with a big item and no small items and bins with small items and no big item do not exist simultaneously. Moreover, if two bins contain small items, one of them has to contain at least $\beta/2$ small items. If such a bin contains also a big item, its size is at least $1/2 + \beta/2 > \alpha$ for our choice of α.

It follows that if a huge or medium item causes a removal of a small item and later a big item would have to be rejected or removed, almost all bins have size at least α and the algorithm can simply stop. (There may be three exceptional bins with size less than α: one bin with a single medium item, one bin with more than β of small items, and one bin with a big item and less than $\beta/2$ of small items.)

If the algorithm exhausts all items before rejecting or removing a big item, we compare the total size of items that were not packed to items that were packed. All items are packed when they arrive and at most size β of small items is removed from each bin (when a medium or huge item is packed there); furthermore, the size of the bin after the step when small items are removed is at least α (as it contains two medium items or one huge item) and thus no further removals occur in that bin. This gives the ratio of packed items to all items of at least $\alpha/(\beta + \alpha) = \alpha$ and the competitive ratio of $1/\alpha$ follows.

Finally, if none of the two cases above applies, a big item is rejected or removed before any small item is removed. It follows that either the algorithm stops with almost all bins of size at least α or else it stops by exhausting the items, in which case all the small items are packed. Then the charging argument given before applies as the packed small items can only improve the ratio.

3.1.4 Proportional case—the full version

We now extend the previous algorithm to the algorithm PROFIT which achieves the competitive ratio of 1.6 for every value of m, not only in the limit. We fix $\alpha = 5/8$, as we have seen that this is the best value already for the asymptotic case.

We follow the general outline of the previous algorithm, but we need to deal with a significant number of special cases. The main obstacle is that when we argue about the total size the algorithm packs, we cannot tolerate a constant number of bins of size less than $5/8$

as we did before. The algorithm needs to continue packing even if there are two or three such bins. Even more difficult is the last such bin: We can fill it greedily to at least $1/2$, but we cannot guarantee more. To deal with this, we typically obtain the desired bound on the total volume by arguing that there is a bin of size at least $6/8$ and averaging its size with the size of the last bin.

However, to guarantee that a bin of size at least $6/8$ is created, we need to modify the algorithm significantly. Most significantly, we try to gain one such bin by packing a big and a medium item together whenever possible. This forces three more changes throughout the algorithm. (1) To guarantee that we have at least one such bin whenever the optimum has one, we need to avoid rejecting or removing big items arbitrarily; instead we guarantee that the smallest big item so far (called the *min-big item*) is never removed or rejected. The min-big item then guarantees that the smallest medium item is packed either with some big item or with another medium item; in both cases we gain a bin with size at least $6/8$ which we can average with the last bin. (2) Another forced change is that we sometimes pack a medium item into a bin that already has size larger than $5/8$, in particular in a bin with a big item and some small items that are removed; in turn we then need to guarantee that in such a bin the size of small items is at most $3/8$, so that no more small items are removed from any single bin. (3) We need to handle huge items of size smaller than $6/8$ more carefully: If such an item should be packed so that it replaces a big item and small items of total size more than $6/8$, it is better to reject the huge item instead. This is a conservative change which can only help the algorithm.

Technically, we describe PROFIT in two main phases. In the first phase, we follow the outline of the simplified algorithm with the changes above as long as no big item is rejected or removed. The description is somewhat lengthy, as this is the place where we need to handle some of the special cases with only a few bins smaller than $5/8$. In the end, if the algorithm ends in this phase, we argue that either the algorithm packs a $5/8$ fraction of the size of all items or it fills $5/8$ of the total volume; the competitive ratio then follows easily.

The second phase starts when each bin has a medium, big or huge item. We again follow the simplified algorithm and try to pack all medium and huge items even at the cost of removing some big items, with the modifications described above. Since there are now fewer types of bins, the description is simpler. If the average size of a bin does not reach $5/8$, we prove the competitive ratio by a charging argument. This is more complicated than in the simplified algorithm, again because we cannot ignore a constant number of bins. First, we cannot assume that all the small items are packed, thus the charging scheme needs to consider the small items as well. Second, if there exists a bin with one medium item (which we have ignored before), its size may be as small as $3/8$, while it can be charged the size of 1; we need to modify the analysis so that such a bin is considered together with one or two other bins.

Preliminaries We classify the items by their sizes as in the simplified algorithm, but with the specific value of $\alpha = 3/8$: an item is *small* if its size is in $[0, 3/8]$, *medium* if it is in $(3/8, 1/2)$, *big* if it is in $[1/2, 5/8)$, and *huge* if it is in $[5/8, 1]$.

For the description and analysis of PROFIT we classify the bins into eight types. At the beginning, we have only **empty bins** with no items. Then we have bins that are not empty and may be used for packing new items. They are of three major types according to the type of the largest item packed into them: **small bins**, **medium bins** and **big bins**. They are further divided into six different subtypes specified as follows:

Regular small bin: A bin with only small items of total size at most $3/8$.

Regular big bin: A bin with a single big item and no other items.

Semi-final big bin: A bin with a single big item and small items of total size at least $1/8$ and at most $3/8$.

Active big bin: A bin with one big item and one or more small items of total size less than $1/8$.

Active medium bin: A bin with one medium item and possibly some small items with total size at most $3/8$.

Active small bin: A bin with only small items of total size more than $3/8$ but less than $5/8$.

As we shall see later, there is always at most one active bin of each major type. The other types of bins can appear in many copies.

The last type of bins are **final bins** that are required to satisfy the following two properties: (1) The size of the bin is at least $5/8$, and (2) small items of size at most $3/8$ were removed from the bin in the step when its type was set to a final bin and no small item was removed before that step. Moreover, once a bin type has been changed to final, no more items are ever packed into it.

Usually the type of the bin is uniquely determined by its contents, except that in some cases the condition of the final bins may be satisfied also by a bin of another type (in particular by a semi-final big bin or a medium bin). Thus the algorithm needs to be explicit in particular in specification of when the bin is declared to be final.

One of the invariants of the algorithm is that no other bins than the types described above ever appear, possibly with the exception of the bin where the last item is packed. The algorithm always specifies the type of a bin whose contents change. In most cases, we leave it to the reader to check that the bin indeed satisfies the conditions of its declared type. In particular, this is the case for the new final bins: verifying that both conditions hold easily follows by inspecting the contents of such a bin. To verify that the condition (2) for the final bin holds, we also use the fact that whenever small items are removed from a bin, either the bin is then declared to be final or the algorithm stops; thus it is sufficient to verify that in the step when the bin is declared to be final, at most $3/8$ of small items are removed, and this typically holds because there are no more small items in the bin before the step.

Algorithm The algorithm stops if (i) there are no more items on the input or (ii) if the average size of all its bins is at least $5/8$.

Otherwise, let a be the item to be packed; we call it the *current item*. The algorithm packs the current item according to the following rules, preferring the first possible option. If multiple bins satisfy the same rule, choose one arbitrarily unless stated otherwise. If the rule allows the removal of the small items, remove the items one by one in an arbitrary order and stop as soon as the current item fits.

At the beginning, the items are processed using the rules for Phase 1. Once no rule for Phase 1 applies, if possible, the algorithm performs a *Terminal Move* consisting in packing the current item so that the average size of all bins is at least $5/8$ and stops. If this is not possible, the algorithm follows the rules for Phase 2 for the current and all later items. First, we present the Phase 1 rules, then we prove some properties holding during Phase 1 and upon entering Phase 2, and only then we finally present the rules of Phase 2.

Phase 1

(1) If the current item a is huge, pack it according to the following options; the bin with a is then always declared to be a final bin:
   (a) Pack a in an empty bin.

(b) If $s_a < 6/8$ and there is one small bin with an item larger than $2/8$ (and possibly some other items), one medium bin with at most $2/8$ of small items, and all the other bins are final or semi-final, perform the *Special Move:* Pack a in the medium bin replacing the medium item.
(c) Pack a in a regular small bin, removing some small items if necessary. Note that if there is no small item of size larger than $2/8$, then either the final bin has size at least $6/8$ or no small items are removed.

(2) If the current item a is medium, pack it according to the following options:
(a) In a big bin if it fits with the big item, possibly removing some small items. The bin is then declared to be final.
(b) In an active medium bin, possibly removing some small items. The bin is then declared to be final.
(c) In an empty bin, which is then active medium.
(d) In an active small bin if it fits. The bin is then declared to be final.
(e) In a regular small bin, which is then active medium.

(3) If the current item a is big, pack it according to the following options:
(a) In an active medium bin if it fits with the medium item, possibly removing some small items; the bin is then declared to be final.
(b) In a regular small bin; it is then semi-final big if the size of small items is at least $1/8$ and it is active big otherwise.
(c) In an empty bin, which is then regular big.

(4) If the current item a is small, pack it according to the following options:
(a) In a regular small bin if its resulting size does not exceed $3/8$; the bin stays regular small.
(b) In an active big bin if the resulting size of the small items in it does not exceed $3/8$. The bin is then semi-final big if the size of small items becomes at least $1/8$, otherwise it stays active big.
(c) In a regular big bin. The bin is then semi-final big if $s_a \geq 1/8$, otherwise it is active big.
(d) In an empty bin, which is then regular small.
(e) In an active medium bin if the item fits. The bin is then declared to be final if its size is at least $6/8$, otherwise it stays active medium.
(f) In an active small bin. The bin is then declared to be final if its size becomes at least $5/8$, otherwise it stays active small.
(g) In the smallest regular small bin. The bin is then declared to be final if its size becomes at least $5/8$, otherwise the bin is active small (its size now exceeds $3/8$).

Terminal Move If none of the previous moves apply, do the following. If it is possible to pack the item so that the average size of all bins is at least $5/8$, do so and terminate the algorithm. Otherwise continue to Phase 2.

The Terminal Move implicitly involves solving a knapsack problem, as for each bin we need to find the set of the items in the bin that fits together with the current item and has the largest size. However, examining the proofs of Lemmata 5, 6, and 9, it turns out that it is sufficient to try to perform the Terminal Move in the following limited form: (i) Put the current item in the medium bin, removing either the medium item or all the items in the bin, or (ii) put the current item in the active big bin, removing the big item and/or all the small items, or (iii) put the current item in the active small bin or one of final bins with only small items, removing all items except the largest one, or (iv) put the current item in the smallest bin and, if necessary, remove the items greedily in the reverse order of their packing until

the item fits. If one of the previous possibilities causes the average size of all bins is at least $5/8$, use it and terminate; otherwise continue to Phase 2.

The first three lemmata summarize simple properties of the algorithm that follow by a simple inspection of its description.

Lemma 2 *At any time during Phase 1, the following holds:*

(i) *Consider all regular small bins and all active bins with small items. For any two bins among them, it holds that the total size of the small items is larger than* $3/8$*. In particular, it follows that for at most one of them the size of small items is at most* $3/16$*.*

(ii) *There is at most one active bin of each kind (medium, small or big).*

Proof (i): For a medium bin with small items, notice that once such a bin is created in step (2e) or (4e), there is no empty or regular big bin and also no such bin can be created later. A small item is put into a bin with no small items only when putting it into any of bins considered in the claim with small items would exceed the size $3/8$ of small items. Thus the claim is maintained whenever a new bin with small items is created.

(ii): A medium bin is created only in Step (2b) when a medium item is packed, but this option always has a smaller priority than putting it in an existing medium bin in Step (2a). Similarly a small item creates an active small bin only in Step (2g) when the same type of bin does not exist, as otherwise Step (2f) would apply. Two big active bins do not exist as the total size of the small items in them would be at most $2/8$, contradicting (i). □

Lemma 3 *If there is a regular big bin or an empty bin or a medium bin with no small item(s), then*

(i) *there is no active small bin, no final bin with only small items, and no active medium bin with small item(s), and*

(ii) *every final bin contains a huge item, or it contains one medium and one big item, or it contains two medium items and no small item was removed from it.*

Proof First note that, since the lemma assumption holds, it held throughout the whole execution of the algorithm until now as no new non-final bin without any small items is ever created. Now (i) follows: an active small bin or a final bin with only small items can be created only in Step (4g) and an active medium bin can be created only in Step (2d) or (4e). However, one of Steps (2a-c) or (4a-d) would have applied first instead. Now (ii) follows: without the bins whose existence is ruled out by (i), any final bin is created either by packing a huge item or by packing medium and big items in one bin, or by packing a medium item into a medium bin with no small item. □

The next two lemmata guarantee that once some condition holds, the algorithm will not enter the second phase. This allows us to significantly restrict the possibilities in the second phase.

Lemma 4 *If the Special Move happened, the algorithm will not enter Phase 2. Furthermore, no item is removed from the only remaining non-final bin unless the algorithm stops in the same move.*

Proof After the Special Move, we are left with a single bin that is neither final nor semi-final; furthermore, this bin has size at least $2/8$. As long as small items arrive, they are put into this bin and no item is removed. When an item of another type comes, it is put in this bin and the size becomes at least $5/8$, even after small items are removed as necessary. So the algorithm stops. □

Lemma 5 *If an active small bin or a final bin with only small items appears at any time during Phase 1, the algorithm will not enter Phase 2.*

Proof We claim that the following hold at *any* moment when an active small bin or a final bin with only small items is created:

(i) There is no regular big bin and no empty bin.
(ii) If a medium bin exists, it has small item(s) and size greater than $5/8$.
(iii) Every regular small bin has size at least $3/16$.

As the bin is created by packing a small item in Step (4g), no rule of higher priority applied. In particular, (i) holds since Steps (4c-d) do not apply, and (ii) holds since Step (4e) does not apply. As for (iii), its first claim follows from Lemma 2(i), which implies that (iii) holds with the possible exception of the smallest regular bin. But as the item we consider is packed into this bin, turning it into an active small bin or a final bin, all remaining regular small bins have size at least $3/16$.

Now we claim that the following, slightly different, properties hold at the end of Phase 1:

(iv) There is no regular big bin and no empty bin. If there is an active big bin, it was present already at the first time when an active small bin or a final bin with only small items was created.
(v) If a medium bin exists, either its size is greater than $5/8$, or the total size of the small items in it is in $[3/16, 2/8]$ and the medium item in it does not fit in the active small bin (if such a bin exists).
(vi) There is no regular small bin.

As any item can be packed in a regular small bin, (vi) follows. The previous two points follow from (i)–(iii). Specifically, (iv) follows from (i): there were no empty or regular big bins and clearly none can be created later. Moreover, any new active big bin would have to be created in Step (3b) from a regular small bin with size less than $1/8$ but there is no such bin due to (iii). As for (v), its first option holds, by (ii), for any medium bin that was present the last time that an active small bin or a final bin with only small items was created. And if the medium bin was created later, it was created by packing a medium item into a regular small bin in Step (2e). As Step (2d) did not apply, the medium item did not fit in the active small bin if there was such a bin. Moreover, the small bin into which the medium item was packed had at least $3/16$ of small items by (iii). We may assume it had no more than $2/8$ of small items as otherwise its total size would exceed $5/8$ and the first option of (v) would apply.

With (iv)–(vi) established, we know that, apart from the final and semi-final bins, only the active bins of each kind may be present; recall that there is at most one active bin of each kind by Lemma 2(ii). Our goal now is to show that since the algorithm did not stop before reaching the end of Phase 1, the Terminal Move can be performed. To this end, we inspect several cases. In each of them, our goal is to show how to pack the current item so that the average size of the active bins is at least $5/8$ since all the other bins are no smaller than $5/8$. For this reason, there must be at least one active bin since otherwise the algorithm would have stopped by now. The cases below are exhaustive: Case 4 covers the case when there is an active big bin, while Cases 1-3 cover the cases when only active small and medium bin or one of them exist.

Case 1: There is an active small bin and either no other active bin or the other active bins have average size at least $5/8$. Put the current item is in the active small bin, removing the small items greedily. This guarantees that the active small bin will have size at least $5/8$ afterward.

Case 2: The only active bin is medium. Then its size is smaller than $5/8$ since otherwise the algorithm would have terminated. Hence, by (v), the total size of the small items in it is in $[3/16, 2/8)$. If the current item is huge, pack it by replacing all the items in the active medium bin. If the current item is big, replace the medium item with it. In both subcases the size of the medium bin is at least $5/8$. There are no more subcases, as a medium or a small element would be packed into the medium bin in step (2b) or (4e).

Case 3: There are both an active small bin and an active medium bin with size smaller than $5/8$. Again, by (v), the total size of the small items in the medium bin is in $[3/16, 2/8)$ and the medium item does not fit in the active small bin. In particular, the active small bin has size at least $4/8$.

If the current item is huge, pack it into the medium active bin as follows: If the current item is larger than $6/8$, remove all other items from the bin, otherwise remove the medium item and keep all the small items. Either way the medium bin now has size greater than $6/8$, so the total size of the two bins is greater than $10/8$. If the current item is big, pack it into the active small bin, removing all the items packed into it in the step when it became active and later. The size of remaining items in the bin is at most $3/8$, so any big item fits. On the other hand, by Lemma 2(i), the total size of remaining small items in both active bins is at least $3/8$. Together with the big and medium items we have at least $3/8+4/8+3/8 = 10/8$ and we reach an average size of $5/8$. There are no more subcases, as a medium or a small element would be packed into the medium bin in step (2b) or (4e).

Case 4: There is an active big bin. In this case we can ignore the active medium bin completely as its size is greater than $5/8$. This follows from the fact that the medium bin (if it exists) has some small items by (v), the fact that the active big bin has less than $1/8$ of small items, and Lemma 2(i).

For convenience, in this case we will refer to to the active big bin as the *big bin*. Similarly, let the *small bin* denote the active small bin if it exists and an arbitrary final bin with only small items otherwise, and let the *smallish item* denote the small item that made the small bin active small or final. Note that due to the lemma assumption, the small bin and the smallish item are well defined. Moreover, as there are no regular small or regular big bins, these names should not cause any confusion.

We begin by observing that the total size of the small items in the small bin and the large bin is greater than $5/8$. To this end, note that the big bin was active big already at the time when the small bin became active small or final by (iv). Then, since the total size of small items in the big bin is less than $1/8$, the size of the small bin right before the smallish item was packed in it was greater than $2/8$ by Lemma 2(i). Moreover, as the smallish item was not packed in the big bin in Step (4b), the sum of its size and that of all small items in the big bin is greater than $3/8$ and together with more than $2/8$ of small items in the small bin, the total size of the small items is at least $5/8$. Also note that each of the two bins has size greater than $4/8$, thus it is sufficient to pack the current item in either bin so that its size is at least $6/8$.

If the current item is huge and larger than $6/8$, it is sufficient to let it replace all items in the active big bin. If the current item is huge but smaller than $6/8$, pack it in the big bin replacing the big item but keeping all the small items; the total size of the big bin and the small bin is now larger than $10/8$ since the total size of the small items in them is greater than $5/8$.

If the current item is medium or big, put it in the small bin together with the smallish item, removing all the other small items in the bin. If the current item is big, this clearly results in a bin of size greater than $6/8$. If the current item is medium, we note that since Step (2a) did not apply, the total size of the current item and the big item in the big bin is

larger than 1. So together with the smallish item, the total size of the two bins is greater than $10/8$.

Finally, if the current item is small, let it replace all the small items in the big bin. Note that the current item's size is greater than $2/8$ since otherwise Step (4b) would be possible; thus the resulting size of the big bin is at least $6/8$. □

The next lemma finally summarizes the situation at the beginning of Phase 2.

Lemma 6 *When the algorithm enters Phase 2, the following holds:*

(i) *No empty, active small or regular small bin exists.*

(ii) *Every final bin contains a huge item, or it contains one medium and one big item, or it contains two medium items and no small item was removed from it.*

(iii) *If a medium bin exists and has one or more small items, no regular or active big bin exists.*

Proof Consider the situation when no option of Phase 1 is possible. We prove that either the Terminal Move is possible or the lemma holds. To prove that the Terminal Move is possible, we only need to show that we can pack the current item so that the average size of the active bins becomes at least $5/8$.

First of all, note that there is no active small bin by Lemma 5. Moreover, there are no empty or regular small bins, as otherwise some Phase 1 move would be possible. This establishes (i).

Since no rule of Phase 1 applies, the current item is not small and, if a medium bin exists, the current item is also not medium.

If there is a regular big bin or a medium bin with no small item(s), both (ii) and (iii) are implied by Lemma 3 as follows: (ii) follows directly from Lemma 3 (ii) and (iii) holds trivially since Lemma 3 (i) implies that there is no medium bin with small item(s).

Thus we can restrict our attention to the case where in addition to final and semi-final bins, we have only an active big bin and/or an active medium bin.

If no active bin exists, the average size is at least $5/8$ and the algorithm would have already terminated. Similarly, if both the active big bin and the active medium bin exist, the algorithm would have already terminated: The total size of small items in them is at least $3/8$ by Lemma 2(i); their average size is thus at least $5/8$.

It remains to consider the case of exactly one active bin.

Suppose that there exists a final bin of size at least $6/8$ and there is only one active bin. Then we need only to attain a size of $4/8$ in the active bin through the Terminal Move. Since the algorithm did not stop before, the active bin has size smaller than $4/8$. In particular, it is a medium bin. As noted before, in such case the current item is big or huge, so even putting it alone in the active medium big would stop the algorithm.

We now inspect the last remaining case in which there is only one active bin and no final bin has size at least $6/8$. Then (iii) holds trivially. As all final bins have sizes strictly smaller than $6/8$, none of them has more than one medium or big item. Hence, to prove (ii) it suffices to show that every final bin contains a huge item. To this end, we show that a final bin contains no small items. By Lemma 5, there is no final bin with only small items and there is no active small bin. Inspecting the rules for big (Step 4) and small (Step 5) items, we see that without items of other kind, these never create a final bin. To rule out a final bin with a medium item and small item(s), note that it is created only in two cases, both ruled out: (2d) because there is no active small bin, and (4e) because all final bins have size strictly smaller than $6/8$. This concludes the proof of the lemma. □

Min-big item. Before describing the packing rules for Phase 2, we define the *min-big item*. At any point, it is the smallest big item that arrived so far in either phase, whether it is currently packed or not. In case of a tie, it is the first such item. Moreover, we say that the min-big item is *compatible* with a medium item if the sum of their sizes is at most 1.

Phase 2

(1) If the current item a is huge, pack it according to the following options:
   (a) Replace all the items of a medium bin with a if this causes the average size of all bins to be at least $5/8$. (This causes the algorithm to stop.)
   (b) Choose any big bin, preferring one which does not contain min-big item.
      – If the chosen bin contains small items of size at least $2/8$ and $s_a \leq 6/8$, we do not pack a and instead we say that a is *assigned* to the chosen big bin for further analysis. No items are removed from the chosen bin. Note that its size is at least $6/8$.
      – Otherwise we pack a in the chosen bin, removing the big item and also the small items in it, if necessary; note that afterwards the bin has size at least $6/8$ or no small item was removed from it.

   The chosen bin is declared to be final in both subcases, i.e., no matter whether a was assigned or packed.

(2) If the current item a is medium, pack it according to the following options:
   (a) In a semi-final, regular big or active big bin if it fits without removing the big item; the small items are removed as necessary and the bin is then declared to be final.
   (b) In a medium bin, possibly removing some small items in it; the bin is then declared to be final.
   (c) In a semi-final, regular big or active big bin; if there are more such bins, choose one without the min-big item and prefer first a regular big bin, then an active big bin. The big item is replaced by a and the bin is then active medium.

(3) If the current item a is big, process it according to the following options:
   (a) Pack a in a medium bin if it fits without removing the medium item but possibly removing some small items; the bin is then declared to be final.
   (b) If a is the min-big item, pack it in a semi-final, active big or regular big bin; a then replaces the big item in the bin and the bin type does not change.
   (c) Otherwise reject a.

(4) If the current item a is small, pack it according to the following options:
   (a) In an active big bin if the resulting size of the small items in it does not exceed $3/8$. The bin is then semi-final if the size of small items becomes at least $1/8$, otherwise it stays active big.
   (b) In a regular big bin. The bin is then semi-final if $s_a \geq 1/8$, otherwise it is active big.
   (c) In a medium bin if the resulting size of the small items in it does not exceed $3/8$. The bin stays active medium.
   (d) In an active big bin or medium bin. We show in Lemma 7(ii that the item fits and the algorithm stops.

As in Phase 1, we first summarize simple properties of the algorithm. Note that the following lemma implies that all the items are processed, which clearly follows from the algorithm description only for big items.

Lemma 7 *At any time during Phase 2, the following holds:*

(i) *There is at most one active medium bin and at most one active big bin. All the other non-final bins are semi-final or regular big.*

(ii) *If Step (4d) is reached, the small item fits in the bin and the algorithm stops afterwards.*
(iii) *All medium items are packed and never removed. All huge items are either packed and never removed, or assigned to some bin. All small items were packed when they arrived and may have been removed only upon creation of a final bin.*
(iv) *If a medium bin contains some small item(s), there is at most one active or regular big bin.*
(v) *Every final bin has a huge item assigned to it, or contains a huge item, or one medium and one big item, or it contains two medium items. If the algorithm removed some small items from a final bin with two medium items, the average size of the bins is at least* $5/8$ *and the algorithm stops.*

Proof (i): As in Phase 1, we pack small items so that each two bins with small items have at least $3/8$ of them, thus there cannot be two active big bins. A medium bin is created only in Step (2c), which can be reached only if no medium bin exists. No other bins exist at the beginning of Phase 2 by Lemma 6 (i) and none are also created later.

(ii) If Step (4d) is reached, there is no regular big bin as Step (4b) did not apply. Thus there exist only final, semi-final and active bins. There must exist at least one active bin, as the size of all the other bins is at least $5/8$ and the algorithm would have stopped otherwise.

We claim that only a single active bin exists. Assume for the sake of contradiction that this is not the case. Then both the active medium bin and the active big bin contain small item(s), as Steps (4a) and (4c) did not apply. Thus the size of small items in the two active bins is at least $3/8$. As there is a big item in the active big bin and a medium item in the active medium bin, the total size of these two active bins is at least $10/8$. Therefore, the average size of all bins is at least $5/8$. This yields a contradiction since the algorithm would have stopped by now.

As there exists only a single active bin, its size is smaller than $5/8$ as the algorithm would have stopped otherwise. So the small item fits. Furthermore, as Steps (4a) and (4c) did not apply, the size of small items is now more than $3/8$, so the size of the bin is more than $5/8$ and therefore the algorithm stops.

(iii): This holds at the beginning of Phase 2 by Lemma 4 and inspecting the rules of Phase 1. In Phase 2, this is maintained trivially for medium items. For a huge item, notice that if no big bin exists, putting the huge item in a medium bin guarantees average size of bins at least $5/8$, so Step (1a) applies. For a small item, this follows from (ii).

(iv): Recall that no new big bin is created in Phase 2. If a medium bin contains some small item(s) at the beginning of Phase 2, there is no regular big bin nor any active big bin by Lemma 6 (iii). If a medium bin is created later, it is created from a big bin by replacing the big item by a medium item. If this big bin is regular, the new medium bin contains no small item. Otherwise, by the preference in Step (2c), any remaining regular or active big bin must contain the min-big element. Thus, there is at most one such big bin.

(v): No final bins other than the ones specified in the claim exist at the beginning of Phase 2 by Lemma 6(ii). It also follows from the rules of Phase 2 that no final bins other than the ones specified are created during Phase 2. For the second part of the claim, note that if a second medium element is put in a medium bin with some small item(s), this bin has size at least $6/8$. Then there is at most one active or regular big bin of size at least $4/8$, and all the other bins have size at least $5/8$, by (iv). Thus, the average bin size is at least $5/8$ and the algorithm stops. □

The following lemma implies that if there are compatible medium and big items, the algorithm creates at least one final bin of size $6/8$.

Lemma 8 *At any time during Phase 2, the following holds:*

(i) *If there exists a medium bin, then the size of the medium item in it plus the size of any big item in a big bin is more than* 1.

(ii) *If the medium item in a medium bin is compatible with the min-big item, the min-big item is packed in a final bin with some medium item.*

Proof (i): It follows because the first option for a medium item is to be packed with a big item in a big bin and the first option for a big item is to be packed with a medium item in a medium bin.

(ii): Suppose for a contradiction that the min-big item is not in a final bin with a medium item and it is compatible with the medium item in the current active medium bin. Thus, when the min-big item arrived, it did not fit with the medium item in the medium bin if any existed at that time. This implies that the current medium bin was created later from some big bin. This in turn implies that the min-big item was packed since at the time of its arrival there was a big bin. As long as there is a big bin, the min-big item is never removed: a big item can be removed only when a huge or a medium item is packed but in both cases the bin with the min-big item is selected only when there are no other big bins. Thus the min-big item is in some big bin when the current active medium bin was created upon packing a medium item. However, as this medium item is compatible with the min-big item, the rules imply that it is packed with some big item, a contradiction. □

The next lemma will later guarantee that the charging scheme is well-defined.

Lemma 9 *At any time during Phase 2, the number of final bins with two medium items is at most the number of big bins.*

Proof If at the beginning of Phase 2 there is a medium bin and there are fewer big bins than final bins with two medium items, the Terminal Move is possible as, even after putting the current item (which is big or huge) into the medium bin instead of the medium item, the number of big bins is at most equal to the number of final bins with two medium items and the average size is thus at least $5/8$. Whenever there is no medium bin, the number of big bins is even strictly larger than the number of final bins with two medium items, or else the average size of a bin is at least $5/8$ and the algorithm stops. Thus the condition holds at the beginning of Phase 2 and also after creating a new active medium bin as this decreases the number of big bins only by 1. The only remaining case when the condition could be violated is when there is the same number of big bins as of final bins with two medium items, there is an active medium bin, and a huge item is placed into a big bin. But in this case, the huge item is placed in the medium bin in Step (1)(a) as the average size becomes at least $5/8$. □

Theorem 4 PROFIT *is* $8/5$*-competitive for the* sum*-objective on proportional instances.*

Proof **Case A.** The algorithm stops because the average size of bins is at least $5/8$. This includes a successful Terminal Move. The theorem follows because the average size of bins in the optimum is at most 1.

Case B. The algorithm exhausted all items during Phase 1. It follows that all medium, big and huge items were packed and none of them was removed with the possible exception of a medium item that was removed in the Special Move. Small items are also all packed but some of them may have been removed when packing medium, big or huge items and making a bin final. However, in each final bin, a volume of at least $5/8$ remains, and a volume of at most $3/8$ small items has been removed from it, with the exception of the Special Move,

which we handle later. Thus at least $5/8$ fraction of the volume of all the items that were packed into the bin remains there, and the ratio $8/5$ follows.

It remains to deal with the Special Move. Upon the Special Move, a final bin from which a medium item is removed is created, and all other bins except one are either final or semi-final. The remaining bin is a small bin with volume at least $2/8$. Moreover, as the algorithm stops by exhausting all items in Phase 1 by the case assumption, no items were removed from that bin. Hence, the total volume of the two bins right after the Special Move is at least $7/8$, and no more than $4/8$ volume (of the removed medium item) was removed from them. This yields ratio $(7+4)/7 < 1.6$ for the pair of bins. All other bins are handled as above.

Case C. The algorithm exhausted all items during Phase 2.

We use a charging argument.

As an auxiliary notion, we define a mapping of optimal bins with medium items to medium items in the algorithm's packing. Provisionally, map each optimal bin to a medium item in it; this is possible since all medium items are packed. Lemma 7(i) implies that there is at most one medium bin. If it exists, m is the medium item in it, and an optimal bin is mapped to m, do the following: (1) If there is another medium item a that has no bin mapped to itself, modify the assignment so that the optimal bin of m is mapped to a. (2) Otherwise, if the optimal bin of m contains a big item but another optimal bin contains a medium item a and no big item, map the bin of a to m and the bin of m to a. (Note that if (1) does not apply, no optimal bin contains two medium items.)

Now we define an assignment of optimal bins to algorithm's bins and pairs of bins. Every optimal bin with a huge item is assigned to the algorithm's bin with the same item or to the bin where the huge item is assigned by the algorithm; this is possible and one-to-one since all huge items are packed or assigned to a final bin. Every optimal bin with a medium item is assigned to a bin with the medium item to which the bin is mapped in the previous paragraph. Every bin of the algorithm that now has two optimal bins assigned is paired with one remaining big bin; this is possible by Lemma 9. The remaining optimal bins (with no medium or huge item) are assigned arbitrarily one-to-one to the remaining bins; this can be done since there is the same number of optimal and algorithm's bins.

The charging is defined as follows. All medium, big and huge items in any optimal bin are charged to the bin where the optimal bin is assigned. The small items are charged to the bin where the algorithm packed them even if they were removed later. We argue then that for each bin and each pair of bins the ratio of charged items to the packed items is at most $8/5$. Most often we argue that the ratio of charged medium, big and huge items plus the removed small items to the packed medium, big and huge items is at most $8/5$. This is sufficient, as adding the items that are packed both in the optimum and in the algorithm can only improve the ratio.

We first prove that the charging above works for all bins and pairs of bins with the exception of the medium bin. In most cases, we ignore the small items packed by the algorithm since considering them can only improve the ratio for each considered pair of bins or bin.

- A pair of algorithm's bins has total size at least $10/8$ as it contains two medium items and a big item; no small items are removed from these bins. The pair of bins is charged items from two optimal bins, a total of at most $16/8$ and the ratio at most $16/10$ follows.
- A final bin with a huge item a is charged at most this huge item or a big item, plus at most $3/8$ of removed small items, so the ratio is at most $(s_a + 3/8)/s_a = 1 + 3/(8s_a) \leq 1 + 3/5 = 1.6$, as $s_a \geq 5/8$.
- A final bin with a huge item assigned is charged this huge item with a size at most $6/8$. As the bin has a big item and no small items were removed from it, the ratio is at most $6/4 = 1.5$.

– A final bin with both medium and big items is charged at most 1 plus $3/8$ for removed small items. Thus, the ratio is at most $(8+3)/(4+3) < 1.6$.
– An unpaired final bin with two medium items is charged at most 1, as no small items are removed from them by Lemma 7 (v), yielding ratio at most $8/6 < 1.6$.
– A big bin is charged at most a big item (possibly different) and no small items are removed, so the ratio is at most $5/4$.

This completes the proof if there is no medium bin.

If there is a medium bin, we need to consider it together with another bin or pair of bins. We pair the medium bin with a final bin with medium and big items combined if such a final bin exists. Otherwise we pair it with any pair of bins (necessarily consisting of a final bin with two medium items and a big bin), creating a triple. If also no pair of bins exists, we pair the medium bin with any final or semi-final bin that either has size at least $6/8$ or no small item was removed from it.

We claim that the pair or triple of the medium bin is well-defined. Any final bin created in Phase 2 satisfies one of the conditions, any final bin with a medium item or a semi-final bin is eligible as well. Thus the only remaining possibility is that all $k-1$ bins other than the medium bin are created in Phase 1 by packing a huge item and removing small items including one larger than $2/8$. This would in turn imply that upon creating the last such bin, we would use the Special Move and the algorithm would end in Phase 1.

To complete the proof, we need to argue that the triple or pair with the medium bin is charged at the correct ratio. We distinguish two cases.

Case C.1 The medium bin is not assigned an optimal bin with both medium and big items.

Then the medium bin has a medium item of size at least $3/8$ and it is charged at most $5/8$. This does not guarantee the ratio, but it is very close and we need to save only a little in the triple or pair of the medium bin. Except for the first two types, we argue as above which works since the ratio was not tight. Calculate depending on where the medium bin is:

– In a triplet with a final bin with two medium items and a big bin we argue as follows: The medium item in the medium bin and the big item in the big bin have total size more than 1 by Lemma 7(v). Together with the two medium items in the final bin, this is at least $14/8$. We are charged at most 2 to the pair of bins and at most $5/8$ to the medium bin. As no small items were removed from either bin by Lemma 7 (v), the ratio is at most $21/14 = 1.5$.
– In a pair with a final bin B with a huge item, we distinguish two options. If the final bin has size $s(B) \geq 6/8$, it is charged at most $s(B)$ for the items in it (including the small ones here) plus at most $3/8$ for removed items. Together with the medium bin we have a ratio of at most $(s(B)+3/8+5/8)/(s(B)+3/8) = 1+5/(8s(B)+3) \leq 1+5/9 < 1.6$, using $s(B) \geq 6/8$. If the final bin has at most $2/8$ of small items removed, we get ratio at most $(5+2+5)/(5+3) = 1.5$. No other option is possible due to the choice of a pair for the medium bin.
– In a pair with a final bin with a huge item assigned. Ignoring the small items, we are charged at most $(6+5)/8$, and we have at least $(4+3)/8$ in medium and big items. Thus the ratio is at most $11/7 < 1.6$.
– In a pair with a final bin with both medium and big items the ratio is at most $(8+3+5)/(4+3+3) = 1.6$.
– In a pair with a final bin with two medium items, the ratio is at most $(8+5)/(6+3) < 1.6$.
– In a pair with a big bin the ratio is at most $(5+5)/(4+3) < 1.6$.

Case C.2 There is a medium bin and it is assigned an optimal bin with one medium and one big item. By the construction of the matching of bins with medium items, it follows that the optimum packs all medium items, and each medium item is packed with a big item.

In this case every medium item fits with the min-big item, thus we know by Lemma 8(ii) that there is also a final bin with medium and big items. Thus the medium bin is paired with one such bin.

We distinguish two more subcases.

Case C.2.1 One of the optimal bins is assigned to a big bin.

Fix one such bin and analyze it together with the pair of the medium bin and a final bin with both medium and big items. The medium item of the medium bin and the big item of the big bin together have size at least 1 as they do not fit together. The final bin has volume at least $7/8$, for a total of at least $15/8$. The triple is charged at most 2 to the final and medium bins, a big item of at most $5/8$ to the big bin, and at most $3/8$ of the removed small items in the final bin. The total charge is at most $24/8$, giving a ratio of at most $24/15 = 1.6$.

Case C.2.2 Each optimal bin with no medium item is assigned to a final bin. (Includes the case when each optimal bin has a medium item.)

In this case we charge the small items differently. We simply charge all the items in the optimal bin, including the small ones, to the assigned bin or pair of bins.

The pair of the medium bin has size at least $3/8$ for the medium bin and $7/8$ for the final bin with both medium and big items, a total of $10/8$; the pair of bins is charged at most 2 from the two optimal bins, the ratio is at most 1.6. Any other pairs of algorithm's bins have total size at least $(4+3+3)/8 = 10/8$ for one big item and two medium ones; it is charged at most 2 from the two optimal bins, the ratio is at most 1.6. All the other bins are charged one-to-one to a final bin of size at least $5/8$, thus the ratio is again at most $8/5 = 1.6$. □

3.2 Lower bounds

Note that proving a $k/(k-1)$ lower bound for deterministic algorithms for the *sum*-objective in the removable variant is straightforward. First issue k items of size 1 and profit 1, and then keep issuing items of size ε^2 and profit ε. Eventually, the gain for those items alone becomes huge, so at some point ALG has to remove one of large items. At that point the ratio is at least $k/(k-1+\varepsilon)$.

Moreover, it follows from known results on *dual bin packing* [1] that even for *randomized* algorithms for the unit case there is a lower bound of $7/6$. Adapting that construction yields a slightly weaker bound for the proportional case.

Theorem 5 *Consider the* sum*-objective in the proportional case and for any $k \geq 2$. The competitive ratio $R_r(k)$ of any randomized k-bin algorithm satisfies*

$$R_r(k) = \begin{cases} \frac{8}{7} & \text{if } k \equiv 0 \\ \frac{8k-2}{7k-1} = \frac{8}{7}\left(1 - \frac{3}{4(7k-1)}\right) & \text{if } k \equiv 1 \end{cases} \pmod 2 \ .$$

The competitive ratio $R_d(k)$ of any deterministic k-bin algorithm satisfies

$$R_d(k) \geq \begin{cases} \frac{8}{7} & \text{if } k \equiv 0 \\ \frac{8t+2}{7t+2} = \frac{8}{7}\left(1 - \frac{1}{2(7k-3)}\right) & \text{if } k \equiv 1 \\ \frac{8t+4}{7t+3} = \frac{8}{7}\left(1 + \frac{2}{7k-2}\right) & \text{if } k \equiv 2 \\ \frac{8t+6}{7t+5} = \frac{8}{7}\left(1 + \frac{1}{7k-1}\right) & \text{if } k \equiv 3 \end{cases} \pmod 4 \ .$$

In particular, $R_d(2) \geq 4/3$, $R_d(3) \geq 6/5$, *and* $R_r(3) \geq 11/10$.

Proof This lower bound is an adaptation of one developed for *dual bin packing* [1], which in turn applies directly to the unit case (see the remark after the proof). The construction is the same for randomized and deterministic algorithms, only the calculation of the bounds at the end is different.

Fix $\varepsilon > 0$ and issue k items of size $1/2 - \varepsilon$. Let K_i ($i = 0, 1, 2$) be the random variable denoting the number of bins into which ALG puts exactly i items. Note that $k = K_0 + K_1 + K_2$ and (w.l.o.g.) $k = K_1 + 2K_2$, as there is no need to remove any item at this point; in particular this implies $K_0 = K_2$.

Let $\alpha := \mathbb{E}[K_0] = \mathbb{E}[K_2]$. We also fix β as follows. If ALG is deterministic, let $\beta := k/4$. If ALG is randomized, let $\beta := k/4$ if k is even and let $\beta = k(k-1)/(4k-1)$ if k is odd. We distinguish two cases based on the order of α and β. In both cases the following operation ends the sequence and OPT packs all items.

1. If $\alpha \geq \beta$, then issue k items of size $1/2 + \varepsilon$. Clearly, $\text{OPT} = k$. Note that all the $K_0 = K_2$ of ALG's bins that held no items until now can now become filled to at most $1/2 + \varepsilon$ capacity each. Hence, $\text{ALG} \leq \left(\frac{1}{2} + \varepsilon\right) K_0 + K_1 + K_2 \leq k - \left(\frac{1}{2} - \varepsilon\right) K_2$. Consequently, for $\varepsilon \to 0$, we get a lower bound on the competitive ratio of $R \geq k/(k - \alpha/2)$. In particular:
   (a) For randomized algorithms and even k as well as for deterministic algorithms and $k = 4t$, we have $\alpha \geq \beta = k/4$ by the case assumption, so the competitive ratio is at least $R \geq k/(k - k/8) = 8/7$.
   (b) For randomized algorithms and odd k, we have $\alpha \geq \beta = k(k-1)/(4k-1)$, so the competitive ratio is at least $R \geq 1/(1 - (k-1)/(8k-2)) = (8k-2)/(7k-1)$.
   (c) For deterministic algorithms and $k = 4t + r$ where $r \in \{1, 2, 3\}$, we note that α is integral, which implies $\alpha \geq t + 1$. Consequently, we obtain:
      i. $R \geq (8t+2)/(7t+1) > (8t+1)/(7t+1)$ for $k = 4t + 1$,
      ii. $R \geq (8t+4)/(7t+3)$ for $k = 4t + 2$, and
      iii. $R \geq (8t+6)/(7t+5)$ for $k = 4t + 3$.
2. If $\alpha \leq \beta$, then issue $\lfloor k/2 \rfloor$ items of size 1. Clearly, for $\varepsilon \to 0$, OPT tends to k if k is even and $k - 1/2$ if k is odd. Note that ALG's best strategy is to put the new items into the bins that had 0, 1, and 2 items, in this order, removing the previous items if necessary. Since $K_0 = K_2$, no bin with two items is actually used and ALG has to remove $\lfloor k/2 \rfloor - K_0$ items of size $1/2 - \varepsilon$.
   (a) For even k and $\varepsilon \to 0$, we have $\mathbb{E}[\text{ALG}] \leq k - \left(\frac{k}{2} - \alpha\right) \cdot \frac{1}{2} = \frac{3}{4}k + \frac{1}{2}\alpha$. Since OPT tends to k, we get $R \geq 4k/(3k + 2\alpha)$. In particular:
      i. For randomized algorithms and for deterministic algorithms in case $k = 4t$, by the case assumption, the competitive ratio is at least $R \geq 4k/(3k + k/2) = 8/7$.
      ii. For deterministic algorithm and $k = 4t + 2$, we have $\alpha \leq t$ and thus $R \geq (8t + 4)/(7t + 3)$.
   (b) For odd k and $\varepsilon \to 0$, $\mathbb{E}[\text{ALG}] \leq k - \frac{1}{2} - \left(\frac{k-1}{2} - \alpha\right) \cdot \frac{1}{2} = \frac{3}{4}k - \frac{1}{4} + \frac{1}{2}\alpha$. Since OPT tends to $k - 1/2$, we have $R \geq (4k-2)/(3k - 1 + 2\alpha)$. In particular:
      i. For randomized algorithms and $k = 2t + 1$, we get $R \geq (4k-2)/(3k - 1 + 2k(k-1)/(4k-1)) = (8k-2)/(7k-1)$.
      ii. For deterministic algorithms, α is integral and hence $\alpha \leq t$. Thus:
         A. $R \geq (8t+1)/(7t+1)$ for $k = 4t + 1$, and
         B. $R \geq (8t+5)/(7t+4) > (8t+6)/(7t+5)$ for $k = 4t + 3$.

□

We remark that this is virtually the same construction as in the original proof for *dual bin packing* [1]. The only difference are the profits of the items: in dual bin packing every item

has profit 1, and thus the problem (roughly) corresponds to our unit case. There are certain differences in the problems as well: in dual bin packing removals are not allowed but one can easily verify that with this construction they do not improve an algorithm's profit. Moreover, in the dual bin packing studies it is usually assumed that OPT packs all items in the instance, which is also the case with this construction. The latter assumption circumvents the $\omega(1)$ lower bound [6], making the problem tractable.

Now we turn our attention to lower bounds for the *max*-objective.

Theorem 6 *No deterministic 2-bin algorithm has a competitive ratio smaller than* 2 *for the* max-*objective.*

Proof The adversary's strategy involves two interwoven sequences of items. To define them, let us fix an arbitrarily small $\varepsilon > 0$. Then the sequences are

small: each item has size ε and profit $\sqrt{\varepsilon}$.
large: the i-th ($i \geq 0$) item has size $1 - \varepsilon \cdot (1 - 2^{-i})$ and profit $1 - i \cdot \sqrt{\varepsilon}$.

Whenever we say that a small (large) item is issued, we mean the successive item from the respective sequence. Initially a small item and a large item are released, in any order. Observe that no large item can be placed together with a small item in one bin, and consequently (w.l.o.g.) ALG puts the large item in one bin and the small one in the other. From this point, the adversary's strategy is as follows: if ALG has a small item in one of its bins, issue a large item, otherwise issue a small item.

Note that by this strategy ALG always has (w.l.o.g.) a large item in one bin, and either another large item or a *single* small item in the other bin; we denote these two states of ALG by LL and LS respectively.

We claim that ALG's ratio is at least $2/(1+2\sqrt{\varepsilon})$, which tends to 2 as ε tends to zero. Assume, for contradiction, it is not so. Then ALG behaves in such a way that no more than $2/\sqrt{\varepsilon}$ small items are issued in total since then the total profit of all the small items would be at least 2, whereas ALG's profit is at most 1. Therefore, eventually ALG loops in the LS state, since a small item is issued every time ALG is in the LL state.

Once ALG loops in the LS state, large items (with slowly decreasing profits) are released in each step, and eventually their profits drop below 0.5. Since $\text{OPT} \geq 1$ due to the first large item and ALG is R-competitive for some $R < 2/(1+2\sqrt{\varepsilon})$, there is a first step when it does not replace its large item with the new one while in the LS state. Denote the large item that ALG keeps by l and the next large one it forfeits by l', and note that $p_{l'} = p_l - \sqrt{\varepsilon} > \frac{1}{2}$ and $s_{l'} < s_l$. Right after l', its "complement" l'' is released with $s_{l''} = 1 - s_{l'}$ and $p_{l''} = p_{l'}$ and the sequence ends. ALG cannot put l'' together with l, so even if it puts l'' together with a small item in the other bin, its profit is at most p_l, whereas OPT's is at least $2p_{l'}$, so the ratio is at least $2/(1+2\sqrt{\varepsilon})$. This is the final contradiction. □

Notice that Theorem 6 matches the upper bound of Theorem 2. However, the former applies to $k = 2$ only. When more than two bins are available, we can only prove the following weaker lower bound.

Theorem 7 *For every $k \geq 3$, no deterministic k-bin algorithm has a competitive ratio smaller than $\phi = (1+\sqrt{5})/2 \approx 1.618$ for the* max-*objective.*

Proof Assume, for the sake of contradiction, that ALG is R-competitive for $R = \phi - \varepsilon$, where $\varepsilon > 0$. Consider the following instance. Initially, two items of size $1/2$ are issued: one of profit 1 and the other of profit ϕ. Since $1 + \phi = \phi^2$ and $R < \phi$, ALG puts both these items into a single bin, say B_1.

Afterwards, a sequence of "small" items of size ε^2 and profit ε is issued until one of the following happens: either the total profit of all the small items issued so far exceeds ϕ^2 or ALG removes an item (w.l.o.g. of profit 1) from B_1.

In the first case, ALG's profits for each of its bins is no larger than $\phi^2+\varepsilon$, while the optimum solution uses the large item of profit ϕ and all the small items for a total profit of at least $\phi+\phi^2=\phi^3$, which gives ratio larger than $R=\phi-\varepsilon$, a contradiction.

Thus we focus on the other case: suppose that the total profit of all the small items released by the moment when ALG removes the item from B_1 is x. At that moment the sequence ends and ALG's profit for any bin other than B_1 is at most x, while the one for B_1 is at most $\phi+\varepsilon$. On the other hand, the optimum solution uses the large item of profit ϕ and, depending on the value of x, either the other large item (of profit 1) or all the small items, whose total profit is x. Hence in this case, the ratio is $(\phi+\max\{1,x\})/\max\{\phi+\varepsilon,x\}$, which is larger than $R=\phi-\varepsilon$ for all $x\leq\phi^2$. □

Theorem 8 (independently proved in [8]) *No randomized 1-bin algorithm has a competitive ratio smaller than* $\frac{e+1}{e}\approx 1.3678$ *in the oblivious adversary model.*

Proof We employ Yao's principle. Fix a large integer n. The set of sequences that we consider are all the prefixes of length larger than 1 of the following sequence of items: an item of size and profit 1, followed by n items of size $1/(n+1)$ and profit $1/n$ each, followed by an item of size $1/(n+1)$ and profit 1.

Observe that, as the first item's size is 1 and the total size of all the remaining items is no larger than 1, every deterministic algorithm (w.l.o.g.) behaves as one the following canonical algorithms. ALG_k keeps the first item (of size and profit 1) in the bin until it sees the k-th small item, i.e., one of size $1/(n+1)$, at which point it removes the large item from its bin and starts collecting the small items. Since, given the chance, w.l.o.g., an algorithm replaces the item of profit 1 and size 1 with the one of same profit but size $1/(n+1)$, we have that $1\leq k\leq n+1$.

With only $n+1$ algorithms to consider, we establish the probability distribution over the $n+1$ instances (recall that the first small item appears in all of them) in such a way that all these algorithms have the same expected profit. Note that this profit is 1 since ALG_{n+1} always holds a single item of profit 1. Let p_i $(1\leq n+1)$ denote the probability of the i-th instance, or, in other words, the probability that the instance ends after the i-th small item.

We fix $\{p_i\}$ as follows

$$p_i=\begin{cases}\frac{1}{n}\left(1-\frac{1}{n}\right)^{i-1}, & \text{for } i\leq n\\ \left(1-\frac{1}{n}\right)^{n}, & \text{for } i=n+1\end{cases} \tag{1}$$

It is straightforward to observe that this is a probability distribution.

As for the expected gains of the algorithms, note that

$$\text{ALG}_k=\sum_{i<k}p_i\cdot 1+\sum_{i=k}^{n}p_i\cdot\frac{i-k+1}{n}+p_{n+1}\cdot\left(\frac{n-k+1}{n}+1\right)\ ,$$

so that for $k\leq n$, the following holds

$$\text{ALG}_{k+1}-\text{ALG}_k=p_k-\sum_{i=k}^{n+1}p_i\cdot\frac{1}{n}=\left(1-\frac{1}{n}\right)\cdot p_k-\frac{1}{n}\cdot\sum_{i=k+1}^{n+1}p_i\ . \tag{2}$$

From (1), it follows that (2) is zero as

$$\frac{1}{n}\cdot\sum_{i=k+1}^{n+1} p_i = \frac{1}{n}\cdot\left(\left(1-\frac{1}{n}\right)^n + \frac{1}{n}\left(1-\frac{1}{n}\right)^k\cdot\sum_{i=0}^{n-k-1}\left(1-\frac{1}{n}\right)^i\right) = \frac{1}{n}\cdot\left(1-\frac{1}{n}\right)^k \quad ,$$

which is exactly $(1-1/n)p_k$.

As every deterministic algorithm's expected gain is at most one, to prove the theorem we only need to lower bound the expected optimum profit. Notice that $\textsc{Opt} = 1$ unless the last item is issued, in which case $\textsc{Opt} = 2$. Thus

$$\mathbb{E}[\textsc{Opt}] = 1 + p_{n+1} = 1 + \left(1-\frac{1}{n}\right)^n \quad ,$$

which tends to $1+1/e$ from below as n tends to infinity. □

Theorem 9 (independently proved in [8]) *No randomized 1-bin algorithm for the proportional case has a competitive ratio smaller than* 1.25 *in the oblivious adversary model.*

Proof We apply Yao's principle. Fix an arbitrarily small $\varepsilon > 0$. Consider the following two instances: in the first one, items of sizes $1/3$ and $2/3+\varepsilon$ are given, in this order, whereas in the second one, these items are followed by a third item of size $2/3$. Each of the two instances occurs with probability $1/2$, or, in other words, after the first two items, the final one is issued with probability $1/2$.

Note that there are only two deterministic online algorithms that we need to consider, determined by their choice when the second item is presented: one of them keeps the item of size $1/3$ and possibly adds the last item of size $2/3$ to it if it appears, while the other removes the item of size $1/3$ to keep the one of size $2/3+\varepsilon$ and does not benefit from the appearance of the third item. Also note that the expected gains of these two algorithms converge to $2/3$ when ε tends to 0: for the first one it is exactly $2/3$ irrespective of ε, whereas for the other it is $2/3+\varepsilon$, the profit of the second item.

To complete the proof, note that, as ε tends to 0, the expected optimum gain tends to $5/6$, and hence the ratio tends to $5/4$. □

We stress again that the Theorems 8 and 9 were proved independently in [8].

4 Non-removable variants

It is well known that no deterministic 1-bin algorithm has constant competitive ratio in the non-removable variant [14,15], even for proportional or unit case. This can be seen by considering (prefixes of) an instance with only two items for the proportional case: one of arbitrarily small size $\varepsilon > 0$, followed by another of size 1. For the unit case, one needs to look at (prefixes of) an instance where an item of size 1 is followed by $1/\varepsilon$ items of arbitrarily small size ε.

We demonstrate that there is a significant difference in these special cases once more than one bin or random bits are available to the algorithm: for each objective function, either of these two advantages allows for an optimum ratio of 2 in the proportional case, but even combined these two advantages are not sufficient to attain constant ratio in the unit case.

4.1 Upper bounds (proportional case)

For the proportional variant, we note that for $k \geq 2$ bins the (deterministic) algorithm FIRSTFIT is 2-competitive for both the *max*-objective and the *sum*-objective, and that it gives rise to a 2-competitive 1-bin randomized algorithm.

Algorithm FIRSTFIT: For each item, put it in the first (the one with lowest number) bin where it fits, ignoring the item if it does not fit in any bin.

Lemma 10 *If $k \geq 2$ bins, then at any time* FIRSTFIT *either has all the items in its bins, or for each $1 \leq i < j \leq k$ we have $p(B_i) + p(B_j) > 1$.*

Lemma 10, whose proof is straightforward, immediately implies the following.

Theorem 10 FIRSTFIT *is 2-competitive for $k \geq 2$ bins for the* sum*-objective on proportional instances.*

Lemma 10 also implies that FIRSTFIT with 2 bins is 2-competitive for the *max*-objective, and simulating one of $\pi(\text{FIRSTFIT}, \{B_1\})$ and $\pi(\text{FIRSTFIT}, \{B_2\})$ chosen uniformly at random constitutes a 2-competitive barely random single-bin algorithm [4].

4.2 Lower bounds

We provide lower bounds for both objective functions for the proportional case. It is known that, for every k, no randomized k-bin algorithm is $O(1)$-competitive in the unit case for the *sum*-objective [6], and it is easy to see that the same follows for the *max*-objective. We give subtler lower bounds for the proportional case: for the *max*-objective, we prove a lower bound of 2 for *randomized* algorithms with *any* number of bins, proving the optimality of FIRSTFIT and its barely random single-bin variant, whereas for the *sum*-objective we only prove a lower bound of $1 + \ln 2 \approx 1.693$, leaving a gap.

Theorem 11 *For every k, no randomized k-bin algorithm has a competitive ratio smaller than 2 for the* max*-objective in the proportional case.*

Proof We use Yao's principle. We consider instances defined as follows. Fix an arbitrarily large integer n and an arbitrarily small $\varepsilon > 0$. For each integer i, $1 \leq i \leq n$, let $s_i = 1/2 + \varepsilon/2^i$. The instance I_ℓ, $1 \leq \ell \leq n$, consists of items of sizes $s_1, s_2, \ldots, s_\ell$, in this order, followed by a single item of size $1 - s_\ell$. Note that in such an instance every item except the last one has size strictly greater than $1/2$, requiring a separate bin, and the last item fits together with the last item of size larger than $1/2$. Therefore, a deterministic k-bin algorithm for such a set of instances, whatever the probability distribution over them, can be (w.l.o.g.) identified by a set of k items, among the potential n items larger than $1/2$, that it is going to put in its k bins given the chance. Note that such an algorithm can have two items in some of the bins iff one of those k items is the last item of size larger than $1/2$, and otherwise it gains at most $1/2 + \varepsilon$ from any bin. By adopting the uniform probability distribution over the n instances, we make the probability of the former at most k/n. Thus with n tending to infinity and ε tending to zero, in the limit the gain of any deterministic algorithm is at most $1/2$ per bin. Clearly, the optimum solution has profit 1 for one of the bins. □

We note that the result of Theorem 11 was already known for $k = 1$ [4] but, unlike ours, the proof technique of [4] does not extend to larger k.

Theorem 12 *For every k, no randomized k-bin algorithm has a competitive ratio smaller than* $1+\ln 2 \approx 1.693$ *for the* sum*-objective in the proportional case.*

Proof The instance consists of up to $n+1$ phases, numbered $0,1,\ldots,n$. In phase j, k items of size (and profit) a_j are released. These sizes satisfy $1/2+\varepsilon = a_0 < a_1 < a_2 < \ldots < a_n = 1$.

Consider an instance with all $n+1$ phases. Denote the expected number of bins that ALG dedicates to the items from the j-th phase ($0 \le j \le n$) by x_j, and note that every item requires a separate bin. W.l.o.g., $x_0 + \ldots + x_n = k$, since it is best to fill any empty bin with an item of size $a_n = 1$.

The adversary may end the instance after any phase, depending on ALG's current expected gain (but not knowing any of ALG's random choices). Notice that if the instance ends after the j-th phase, then clearly $\text{OPT} = k \cdot a_j$. When ALG is considered, its expected number of bins used for items from the i-th phase ($0 \le i \le j$) would clearly be x_i. Hence, by ending the instance after the j-th phase, the adversary would force a competitive ratio of at least

$$R_j = k \cdot a_j \cdot \left(\sum_{i=0}^{j} a_i \cdot x_i \right)^{-1} \quad .$$

In this strategy the adversary first chooses a vector $\mathbf{a}$, then ALG chooses a vector $\mathbf{x}$, and finally the adversary chooses j. Hence, the ratio that the adversary can enforce is

$$\max_{\mathbf{a}} \min_{\mathbf{x}} \max\{R_0, R_1, \ldots, R_n\} \quad , \tag{3}$$

and thus, given $\mathbf{a}$, ALG's best strategy is to choose $\mathbf{x}$ that minimizes the inner maximum. Of course, the vectors $\mathbf{x}$ that we consider have only non-negative entries that sum up to k.

We claim, cf. Lemma 11, that the inner maximum in (3) is minimized when an $\mathbf{x}$ is chosen such that all the ratios R_j are equal. Hence, w.l.o.g., ALG chooses $\mathbf{x}$ that equalizes all these ratios. We denote this common ratio by R, and note that $R = R_0 = k/x_0$. We further note that R equals each of the following, for $j = 1,\ldots,n$,

$$R'_j = k \cdot \frac{a_j - a_{j-1}}{a_j \cdot x_j} \quad ,$$

where R'_j is the ratio of the increases of the optimum to algorithm's gain between phases $j-1$ and j.

Equalizing $R_0 = k/x_0$ with each R'_j, we get:

$$\begin{aligned} x_0 &= \frac{k}{R} \\ x_j &= \frac{k}{R} \cdot \left(1 - \frac{a_{j-1}}{a_j}\right) \text{ for } j > 0 \quad . \end{aligned}$$

We have to ensure that the x's sum to k, which is equivalent to

$$R = 1 + \sum_{j=1}^{n} \left(1 - \frac{a_{j-1}}{a_j}\right) = n + 1 - \sum_{j=1}^{n} \frac{a_{j-1}}{a_j} \quad .$$

Notice that, by the mean inequality,

$$\sum_{j=1}^{n} \frac{a_{j-1}}{a_j} \ge n \cdot \sqrt[n]{\frac{a_0}{a_n}} = n \cdot \sqrt[n]{\frac{1}{2}+\varepsilon} \quad .$$

Thus with ε tending to 0 (for any fixed n), $R = 1 + n\left(1 - \sqrt[n]{\frac{1}{2}}\right)$ is the maximum we can force, attained when a forms a geometric sequence (with $a_0 = 1/2 + \varepsilon$ and $a_n = 1$ fixed).

It is easy to verify, with e.g. de l'Hôpital's rule, that

$$\lim_{n\to\infty} n\left(1 - \sqrt[n]{\frac{1}{2}}\right) = \ln 2 \ ,$$

which concludes the proof of the theorem. □

Lemma 11 *The inner maximum in* (3) *is minimized when such an* $\mathbf{x}$ *is chosen that all the ratios* R_j *are equal.*

Proof To prove this, it is convenient to focus on the reciprocals of R_j's, i.e.,

$$R_j^{-1} = \frac{\sum_{i=0}^{j} a_i \cdot x_i}{k \cdot a_j} \ , \tag{4}$$

in which case the goal is to maximize their minimum.

Assume for the sake of contradiction that the R_j^{-1} values are not all equal, and let j_0 and j_1 be the "leftmost" (i.e., minimum) and "rightmost" (i.e., maximum) j for which $R_{j_0+1}^{-1}$ attains minimum.

We begin by eliminating simple special cases, namely of that of $j_0 > 0$ and that of $j_1 < n$.

Suppose that $j_0 > 0$. Then, as $R_0^{-1} > R_{j_0}^{-1} \geq 0$, it follows from (4) that $x_0 > 0$. In that case, decrease x_0 by ε and increase x_{j_0} by the same amount. It follows from (4) that R_j^{-1} increases for all $j \geq j_0$ and decreases for all $j < j_0$. But as all the minima of R_j^{-1} are attained for $j \geq j_0$, ε can be chosen so that the minimum is still attained at j_1. This yields a contradiction, since we have increased $R_{j_1}^{-1}$.

The case of $j_1 < n$ is symmetrical. We first note that as $R_0^{j_1} < R_{j_1+1}^{-1}$, we have $x_{j_1+1} > 0$ since the denominator of (4) is strictly greater for $j_1 + 1$. Thus we can decrease x_{j_1+1} by ε and increase x_{j_0} by the same amount. This does not affect R_j^{-1} for $j < j_0$, increases it for j such that $j_0 \leq j \leq j_1$, and decreases it for $j > j_1$. However, as all the minima of R_j^{-1} are attained for $j_0 \leq j \leq j_1$, ε can be chosen so that the minimum is still attained at j_1. This yields a contradiction since we have increased $R_{j_1}^{-1}$.

Thus it remains to prove the case of $j_0 = 0$ and $j_1 = n$. Then let j_2 be minimum such that $R_{j_2}^{-1} > R_0^{-1}$. Since in particular $R_{j_2}^{-1} > R_{j_2-1}^{-1}$, as before we know that $x_{j_2} > 0$. We now reduce this case to the previous one as follows. Decrease x_{j_2} by ε and increase x_{j_2+1} by the same amount. Clearly, R_j^{-1} does not change for $j < j_2$, it increases for $j > j_2$, and decreases for $j = j_2$. Thus the minimum is no longer attained at $j_1 = n$. Moreover, for sufficiently small ε, we maintain that $R_{j_0}^{-1} \leq R_{j_2}^{-1}$, i.e., a minimum of unchanged value is still attained at $j_0 = 0$. This completes the reduction to the second case and the proof. □

5 A note on the adaptive adversary

We note that the observations of Section 2.1 need not hold in the adaptive adversary model. Namely, as such an adversary solves an instance on-line, it may benefit from the presence of additional bins even under the *max*-objective. Thus it is possible that the optimum competitive ratio in such a setting is not a non-increasing function of k. However, for $k = 1$, we establish tight bounds for randomized algorithms in the adaptive adversary model by proving that they cannot perform better than deterministic algorithms.

Theorem 13 *In the adaptive adversary model, no randomized 1-bin algorithm is $O(1)$-competitive for the general case of the removable variant.*

Proof We first describe how the adversary creates the instance, and afterwards specify ADV, the adversary's online algorithm for solving instances created this way.

Fix a large integer n. In the beginning, the adversary issues a single item of size and profit 1, which is followed by a sequence of "small" items of size n^{-2} and profit n^{-1} each. This sequence ends as soon as ALG removes the big item from its bin (note that ALG packs it w.l.o.g.) or n^2 small items have been released so far.

Since the instance ends as soon as ALG removes the large item from its bin, ALG is completely defined by the sequence of (conditional) probabilities $p_1, p_2, \ldots, p_{n^2}$, where p_k is the (conditional) probability that ALG keeps the large item in its bin when it is presented with the k-th small item. Notice that

$$\mathbb{E}[\text{ALG}] = \prod_i p_i \cdot 1 + \left(1 - \prod_i p_i\right) \cdot \frac{1}{n} \quad . \tag{5}$$

Given the sequence $p_1, p_2, \ldots, p_{n^2}$, we define ADV as follows: if $\prod_i p_i \leq 1/n$, ADV keeps the large item forever, otherwise ADV removes it to accommodate the first small item and collects all the small items in the instance.

We examine the ratio of ADV's and ALG's expected gains in each case. If $\prod_i p_i \leq 1/n$, then ADV $= 1$ irrespective of ALG's actual choices, and $\mathbb{E}[\text{ALG}] \leq 2/n$ by (5), yielding a ratio of at least $n/2$. On the other hand, if $\prod_i p_i > 1/n$, then ADV's expected gain can be lower bounded by $n \cdot \prod_i p_i$ since, with probability $\prod_i p_i$, ADV collects all the small items. Combining this with (5), we get

$$\frac{\mathbb{E}[\text{ADV}]}{\mathbb{E}[\text{ALG}]} \geq \frac{n \cdot \prod_i p_i}{\frac{1}{n} + (1 - \frac{1}{n}) \prod_i p_i} \geq \frac{n}{2} \quad ,$$

where the inequality holds because the left hand side increases with $\prod_i p_i$.

Hence, in both cases, the ratio is at least $n/2$. The theorem follows by taking an arbitrarily large n. □

Theorem 14 *In the adaptive adversary model, no randomized 1-bin algorithm for the proportional case of the removable variant has a competitive ratio smaller than $\phi = (1 + \sqrt{5})/2 \approx 1.618$.*

Proof We emulate the adversary's strategy against deterministic algorithms from [10]. Fix an $\varepsilon > 0$ and an integer n. Let a denote an item of size ϕ^{-1} and b denote an item of size $\phi^{-2} + \varepsilon$. As $\phi^{-1} + \phi^{-2} = 1$, the items a and b do not fir together in a single bin.

The adversary uses only items of type a and b (with the possible exception of the last step), issuing either of these in the very beginning, and tries to reach a *winning* state: one in which ALG has an item of type a in its bin while ADV has an item of type b or vice versa. If a winning state is reached, the instance terminates in one of the ways we describe later. Otherwise the game is in a *choice* state: both players have an item of the same kind (either a or b) in their bin.

The first choice state occurs when the second item is issued. In general, in the i-th choice step, as long as $i \leq n$, the adversary issues an item of the kind that players do not have in their bins. ADV does the following: if ALG replaces its current item with the new one with probability at least $1/2$, then ADV keeps his current item, otherwise ADV replaces it with

the new one. Observe that with such a strategy, the game moves directly from a choice state to a winning state with probability at least $1/2$. If the $(n+1)$-st choice step is reached, the adversary ends the instance immediately by not issuing any further items; in such a case, both ADV and ALG keep the common item they had in their bins.

If a winning state is reached, the following happens, depending on who has which type of item:

1. ALG has item a while ADV has item b. Then the adversary issues an item of size $\phi^{-1}-\varepsilon$ and ends the instance: ADV adds the item to its bin, and hence $\text{ADV}=1$, whereas $\text{ALG}\leq\phi^{-1}$.
2. ALG has item b while ADV has item a. Then the adversary ends the instance immediately without issuing further items: $\text{ADV}=\phi^{-1}$, while $\text{ALG}=\phi^{-2}+\varepsilon$.

In both these cases, with ε tending to 0, the total size of ADV's items is at least $1/\phi$ and it is ϕ times larger than the size of ALG's item. As any player's gain in any state is bounded by 1 and the probability that a winning state is not reached after n choice steps is at most 2^{-n}, with n tending to infinity, the ratio of ADV's to ALG's expected gain tends to ϕ. □

Theorem 15 *In the adaptive adversary model, no randomized 1-bin algorithm is $O(1)$-competitive for the proportional case of the non-removable variant.*

Proof Let ALG be a randomized 1-bin algorithm. To prove the theorem, for every $\varepsilon>0$, we give an adversary's strategy such that ALG's expected gain is at most an ε fraction of ADV's expected gain. We specify the way the adversary creates and solves the instance in an online manner at the same time.

Initially, the adversary issues an item of size ε. Let p be the probability that ALG puts this item in its bin.

If $p\leq\varepsilon$, then the adversary puts the item in its bin and ends the instance. In this case, ADV's (expected) gain is ε, whether ALG does put the item in its bin or not. But ALG's expected gain is only a $p\leq\varepsilon$ fraction of that.

Otherwise, ADV does not put the item in its bin. Next, if ALG did eventually put the item in its bin, the adversary issues an item of size 1. In that case, ADV puts it in its bin while ALG cannot do that. Hence, in this case, the expected gains of ADV and ALG are $p\cdot 1$ and $p\cdot\varepsilon$ respectively, so their ratio is at least $1/\varepsilon$ again. □

6 Conclusion and open problems

A single gap remains in the non-removable variant for the *sum*-objective in the proportional case. The gaps for this objective in the removable variant are also significant, both in the general and the two special cases that we studied.

There are many gaps to be bridged in the removable variant under the *max*-objective as well. Of those, the question whether more than 2 bins or randomization permit ratios smaller than 2 in the general case seems particularly interesting. A related direction of interest is relating the power of barely random algorithms to unrestricted randomized algorithms.

Acknowledgements

We thank Monaldo Mastrolilli for suggesting the study of knapsack problems, Fabrizio Grandoni for suggesting the study of the sum objective, Yann Disser for helpful discussions, and anonymous referees for their many useful comments.

Marek Cygan was partially supported by the Polish National Science Center (NCN) grant N N206 567940. Łukasz Jeż was partially supported by the Israeli Centers of Research Excellence (I-CORE) program, Center No.4/11, Foundation for Polish Science (FNP) START Scholarship, and the Polish National Science Center (NCN) grant DEC-2013/09/B/ST6/01538. Jiří Sgall was partially supported by project 14-10003S Czech Science Foundation (GA ČR).

References

1. Y. Azar, J. Boyar, L. Epstein, L. M. Favrholdt, K. S. Larsen, and M. N. Nielsen. Fair versus unrestricted bin packing. *Algorithmica*, 34(2):181–196, 2002.
2. Y. Azar and E. Khaitsin. Prompt mechanism for ad placement over time. In *Proc. of the 4th Int. Symp. on Algorithmic Game Theory (SAGT)*, pages 19–30, 2011.
3. S. Ben-David, A. Borodin, R. M. Karp, G. Tardos, and A. Wigderson. On the power of randomization in online algorithms. *Algorithmica*, 11(1):2–14, 1994. Also appeared in *Proc. of the 22nd ACM Symp. on Theory of Comput. (STOC)*, pages 379–386, 1990.
4. H.-J. Böckenhauer, D. Komm, R. Královic, and P. Rossmanith. On the advice complexity of the knapsack problem. In *Proc. of the 10th Latin American Theor. Informatics Symp. (LATIN)*, pages 61–72, 2012.
5. A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.
6. J. Boyar, L. M. Favrholdt, K. S. Larsen, and M. N. Nielsen. The competitive ratio for on-line dual bin packing with restricted input sequences. *Nord. J. Comput.*, 8(4):463–472, 2001.
7. C. Chekuri and I. Gamzu. Truthful mechanisms via greedy iterative packing. In *Proc. of the 12th Int. Workshop on Approx. Algorithms for Comb. Optim. (APPROX)*, pages 56–69, 2009.
8. X. Han, Y. Kawase, and K. Makino. Randomized algorithms for removable online knapsack problems. In *Proc. of the 3rd Joint Int. Conf. on Frontiers in Algorithmics and Algorithmic Aspects in Information and Management (FAW-AAIM)*, pages 60–71, 2013.
9. X. Han, Y. Kawase, K. Makino, and H. Guo. Online removable knapsack problem under convex function. *Theor. Comput. Sci.*, 540:62–69, 2014.
10. K. Iwama and S. Taketomi. Removable online knapsack problems. In *Proc. of the 29th Int. Colloq. on Automata, Languages and Programming (ICALP)*, pages 293–305, 2002.
11. K. Iwama and G. Zhang. Online knapsack with resource augmentation. *Information Processing Letters*, 110(22):1016–1020, 2010.
12. B. Kalyanasundaram and K. Pruhs. Maximizing job completions online. *J. of Algorithms*, 49(1):63–85, 2003. Also appeared in *Proc. of the 6th European Symp. on Algorithms (ESA)*, pages 235–246, 1998.
13. H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack problems*. Springer, 2004.
14. G. S. Lueker. Average-case analysis of off-line and on-line knapsack problems. *J. Algorithms*, 29(2):277–305, 1998. Also appeared in *Proc. of the 6th ACM-SIAM Symp. on Discrete Algorithms* (SODA'95).
15. A. Marchetti-Spaccamela and C. Vercellis. Stochastic on-line knapsack problems. *Math. Program.*, 68:73–104, 1995.
16. S. Martello and P. Toth. *Knapsack problems*. John Wiley & Sons, 1990.
17. J. Noga and V. Sarbua. An online partially fractional knapsack problem. In *Proc. of the 8th Int. Symp. on Parallel Architectures, Algorithms, and Networks (ISPAN)*, pages 108–112, 2005.