Improved Lower Bounds for the Online Bin Packing Problem with Cardinality Constraints

Hiroshi Fujiwara* Koji Kobayashi

Abstract

The bin packing problem has been extensively studied and numerous variants have been considered. The *k-item bin packing* problem is one of the variants introduced by Krause et al. in Journal of the ACM 22(4). In addition to the formulation of the classical bin packing problem, this problem imposes a *cardinality constraint* that the number of items packed into each bin must be at most k. For the *online* setting of this problem, i.e., the items are given one by one, Babel et al. provided lower bounds $\sqrt{2} \approx 1.41421$ and 1.5 on the asymptotic competitive ratio for $k = 2$ and 3, respectively, in Discrete Applied Mathematics 143(1-3). For $k \geq 4$, some lower bounds (e.g., by van Vliet in Information Processing Letters 43(5)) for the online bin packing problem, i.e., a problem without cardinality constraints, can be applied to this problem.

In this paper we consider the online k-item bin packing problem. First, we improve the previous lower bound 1.41421 to 1.42764 for $k = 2$. Moreover, we propose a new method to derive lower bounds for general k and present improved bounds for various cases of $k \geq 4$. For example, we improve 1.33333 to 1.5 for $k = 4$, and 1.33333 to 1.47058 for $k = 5$.

1 Introduction

The bin packing problem is a classical problem in the field of computer science, which has been most extensively studied. This problem is defined as follows. We are given a sequence of *items*, each of which has a *size* in $(0, 1]$, as an input, and an infinite number of *bins*. Each item has to be packed into one of the bins, and the sum of sizes of items packed into each bin has to be at most one. A bin that contains at least one item is said to be *non-empty*. The goal of this problem is to minimize the number of non-empty bins.

The bin packing problem has been studied also in the *online* setting: The items are given one by one, and each item has to be packed before the next one is given. This problem is quite important in both theoretical and applied aspects, and much work has been done on this problem (e.g. [RBLL89, vV92, Sei02, BBG12]). Online algorithms are usually evaluated using competitive analysis [BE98, ST85]. For any sequence σ of items, and any algorithm ALG, let $C_{ALG}(\sigma)$ denote the number of ALG's non-empty bins for σ. Let OPT be an optimal offline algorithm. Then, for any online algorithm ON, define $R_{ON} = \limsup_{n\to\infty} \sup_{\sigma}\{C_{ON}(\sigma)/C_{OPT}(\sigma) \mid C_{OPT}(\sigma) = n\}$, which we call the *asymptotic competitive ratio* (also known as the *asymptotic performance ratio*) of ON.

A constraint that the number of items packed into one bin is somehow restricted seems quite realistic in application. For example, there exists the minimum size of files used by a computer, and the number of files stored on the computer is thus bounded. In light of this situation, Krause et al. [KSS75, KSS77] introduced the *k-item bin packing* problem, in which the *cardinality constraint* that each bin can contain at most k items is imposed. (They defined

*This work was supported by KAKENHI (23700014 and 23500014).

Table 1: Previous results and our results for the online k-item bin packing problem.

k	lower bound	upper bound
2	$\sqrt{2}(\approx 1.41421)$ [BCKK04] $\rightarrow$ 1.42764 [this paper]	$1+\frac{\sqrt{5}}{5}(\approx 1.44721)$ [BCKK04]
3	1.5 [BCKK04]	1.75 [Eps06]
4	$\frac{4}{3}(\approx 1.33333)$ [vV92] $\rightarrow$ 1.5 [this paper]	$\frac{71}{38}(\approx 1.86843)$ [Eps06]
5	$\frac{4}{3}(\approx 1.33333)$ [vV92] $\rightarrow$ $\frac{25}{17}(\approx 1.47058)$ [this paper]	$\frac{771}{398}(\approx 1.93719)$ [Eps06]
6	1.5 [Yao80]	$\frac{287}{144}(\approx 1.99306)$ [Eps06]
7 to 9	1.5 [Yao80]	2 [BCKK04]
10 to 41	1.5 [Yao80] $\rightarrow$ (See Table 2 in Section 3.)	
42 to 293	$\frac{217}{141}(\approx 1.53900)$ [vV92]	
∞	$\frac{248}{161}(\approx 1.54037)$ [BBG12]	1.58889 [Sei02]

this problem as a scheduling problem.) This problem has been well studied in both the offline and online settings.

Previous Results and Our Results. In the *online k-item bin packing* problem, in which items are given in an online manner and the number of items in a bin has to be at most k, Babel et al. [BCKK04] showed that for $k = 2$, the asymptotic competitive ratio of any online algorithm is at least $\sqrt{2} \approx 1.41421$. Also, they presented a lower bound of 1.5 when $k = 3$ using the method by Yao [Yao80]. Moreover, for larger k, various lower bounds by van Vliet [vV92], Yao [Yao80], and Balogh et al. [BBG12] for the online bin packing problem, i.e., a problem without cardinality constraints, can be applied to the online k-item bin packing problem. We mention that the lower bounds for $k = 4$ and 5 are straightforwardly given by manipulating the method in [vV92]. (See Table 1.)

In this paper, we consider the online k-item bin packing problem. First, we show that the asymptotic competitive ratio of any algorithm is at least $\overline{r} \approx 1.42764$ for $k = 2$, where $\overline{r}$ is the root of the equation $2r^3 - 17r^2 + 30r - 14 = 0$ between $\frac{4}{3}$ and $\frac{3}{2}$, which improves the previous lower bound. Second, we extend the method to obtain lower bounds for the online bin packing problem by van Vliet [vV92] and get various improved lower bounds for various cases of $k \geq 4$. For example, we improve 1.33333 to 1.5 for $k = 4$, and 1.33333 to 1.47058 for $k = 5$. (See Table 1, and Table 2 in Section 3.)

Related Results. In the online k-item bin packing problem, Krause et al. [KSS75, KSS77] showed that for any k, the asymptotic competitive ratio of the most basic algorithm FirstFit is at most $2.7 - 12/5k$. Babel et al. [BCKK04] established an algorithm whose asymptotic competitive ratio is at most 2 for any k. Moreover, Babel et al. [BCKK04] and Epstein [Eps06] designed algorithms for small k. These results are also presented in Table 1. In addition, Epstein [Eps06] established a bounded space algorithm. She showed that its asymptotic competitive ratio is at most 2.69104, and is asymptotically optimal. Note that while a bounded space algorithm always has only a constant number of bins available to accept items, the other results described above, including our new results, focus on unbounded space algorithms. There are some studies [KP99, CKP03, EL10] about approximation algorithms for the k-item bin packing problem. Needless to say, the online bin packing problem (without cardinality constraints) has

been much studied, and the best upper and lower bounds are 1.58889 by Seiden [Sei02] and $248/161 \approx 1.54037$ by Balogh et al. [BBG12], respectively.

2 A Lower Bound for $k = 2$

In this section we present a lower bound of 1.42764 for $k = 2$. We first define an adversary, which determines the size of the next item adaptively according to the behavior of an online algorithm. The strategy of the adversary is chosen from the three strategies whose pseudocodes will be later given as ROUTINE1, 2, and 3, respectively.

We begin by mentioning three subroutines called by ROUTINE1, 2, and 3. See their pseudocodes SUBROUTINE1, 2, and 3 below. Roughly speaking, each subroutine gives a sequence of items while changing the size within a specified range. The only difference between them is just the termination conditions. Let us see the details. Each subroutine is called with four parameters: an online algorithm ON and three values `Min`, `Max`, and `Length` with `Min` $<$ `Max`. Each subroutine returns the current value of the internal variable `tmpMin`. The sizes of given items lie in (`Min`, `Max`). For ease of presentation, if an algorithm ALG is about to put an item into a bin that contains no item, we say that ALG *opens* the bin. The termination conditions of the three subroutines are as follows: SUBROUTINE1 finishes when it has given `Length` items to ON, SUBROUTINE2 finishes when ON has opened new `Length` bins, and SUBROUTINE3 finishes when ON has created `Length` bins with two items.

Before giving their pseudocodes, we define the function f used in these subroutines: for any $x, y \in (0, 1]$ with $x < y$, $f(x, y) = (x + y)/2$. (Indeed, f can be any function that maps x and y to a value between x and y.)

SUBROUTINE1(ON, `Min`, `Max`, `Length`):

Step 1. $a_1 := f($`Min`, `Max`$)$, `tmpMax` := `Max`, `tmpMin` := `Min`, and $i := 1$.
Step 2. Give an item b_i of size a_i, and do the following according to ON's action.
  Case 2.1. If ON opens a bin and puts b_i into it,
    $a_{i+1} := f($`tmpMin`$, a_i)$ and `tmpMax` $:= a_i$.
  Case 2.2. Otherwise,
    $a_{i+1} := f(a_i,$ `tmpMax`$)$ and `tmpMin` $:= a_i$.
Step 3. If `Length` $= i$, then return `tmpMin`. Otherwise, $i := i + 1$, and go to Step 2.

SUBROUTINE2(ON, `Min`, `Max`, `Length`):

Step 1. $a_1 := f($`Min`, `Max`$)$, `tmpMax` := `Max`, `tmpMin` := `Min`, and $i := 1$.
Step 2. Give ON an item b_i of size a_i, and do the following according to ON's action.
  Case 2.1. If ON opens a bin and puts b_i into it, then
    $a_{i+1} := f($`tmpMin`$, a_i)$ and `tmpMax` $:= a_i$.
  Case 2.2. Otherwise,
    $a_{i+1} := f(a_i,$ `tmpMax`$)$ and `tmpMin` $:= a_i$.
Step 3. If the number of bins that were opened by ON at Case 2.1 is `Length`, then return `tmpMin`. Otherwise, $i := i + 1$, and go to Step 2.

SUBROUTINE3(ON, `Min`, `Max`, `Length`):

Step 1. $a_1 := f($`Min`, `Max`$)$, `tmpMax` := `Max`, `tmpMin` := `Min`, and $i := 1$.
Step 2. Give ON an item b_i of size a_i, and do the following according to ON's action.
  Case 2.1. If ON opens a bin and puts b_i into it, then
    $a_{i+1} := f($`tmpMin`$, a_i)$ and `tmpMax` $:= a_i$.

Case 2.2. Otherwise,
$a_{i+1} := f(a_i, \texttt{tmpMax})$ and $\texttt{tmpMin} := a_i$.

Step 3. If the number of bins with two items both of which are given at Step 2 is `Length`, then return `tmpMin`. Otherwise, $i := i + 1$, and go to Step 2.

The purpose of these subroutines is to construct a sequence that has the following property. The proof of the lemma will be provided in the full version.

Lemma 1. *Suppose that* SUBROUTINE1 *(*SUBROUTINE2*,* SUBROUTINE3*, respectively) is called with some ON,* `Max`*, and* `Min` *with* $\texttt{Max} > \texttt{Min}$*. Let β_j be the size of the $j(= 1, \ldots, n)$-th item that is put into a bin at Case 2.1, and let $\gamma_{j'}$ be the size of the $j'(= 1, \ldots, m)$-th item that is put into a bin at Case 2.2. Denote $\beta_0 :=$* `Max`*, $\gamma_0 :=$* `Min`*, and t_{min} (t_{max}) the value of* `tmpMin` *(*`tmpMax`*, respectively) at the moment when the subroutine returns. Then, $\beta_0 > \beta_1 > \cdots > \beta_n = t_{max} > t_{min} = \gamma_m > \cdots > \gamma_1 > \gamma_0$.*

Now we are ready to describe the main routines any of which the adversary chooses as its strategy. We remark here that ROUTINE1 outputs an equivalent sequence to one used for getting a lower bound for $k = 2$ in [BCKK04]. In that analysis the competitiveness of an online algorithm depends on how it packs the items that correspond to Step 1. Our analysis, in addition, examines how to deal with the items given in Step 3 and Step 4 of ROUTINE2 and 3.

The variables t, b, s, x, y, u, z, w, and v appearing in the pseudocodes are used both for the execution of the routine and for the later analysis. "#" stands for "the number of".

ROUTINE1(ON, `Length`):

Step 1. Call SUBROUTINE1(ON, $\frac{1}{10}$, $\frac{1}{9}$, `Length`), and $t :=$ (the return value).
Then, $\frac{x}{2} :=$ (# bins with two items), and $y :=$ (# bins with exactly one item).

Step 2. Give ON $\frac{x}{2}$ items of size $1 - t$.

ROUTINE2(ON, `Length`):

Step 1. Call SUBROUTINE1(ON, $\frac{1}{10}$, $\frac{1}{9}$, `Length`), and $t :=$ (the return value).
Then, $\frac{x}{2} :=$ (# bins with two items), and $y :=$ (# bins with exactly one item).

Step 2. Give ON $\frac{x}{2}$ items of size $1 - t$.

Step 3. Call SUBROUTINE2(ON, $\frac{4}{5}$, $\frac{7}{8}$, $y + \frac{x}{2}$), and $b :=$ (the return value).
Then, $u :=$ (# bins with one item given in Step 1 and one given in Step 3).

Step 4. Call SUBROUTINE1(ON, $\frac{1}{6}$, $1 - b$, u).
Then, $z :=$ (# bins with one item given in Step 1 and one given in Step 4).

ROUTINE3(ON, `Length`):

Step 1. Call SUBROUTINE1(ON, $\frac{1}{10}$, $\frac{1}{9}$, `Length`), and $t :=$ (the return value).
Then, $\frac{x}{2} :=$ (# bins with two items), and $y :=$ (# bins with exactly one item).

Step 2. Give ON $\frac{x}{2}$ items of size $1 - t$.

Step 3. Call SUBROUTINE2(ON, $\frac{4}{5}$, $\frac{7}{8}$, $y + \frac{x}{2}$), and $b :=$ (the return value).
Then, $u :=$ (# bins with one item given in Step 1 and one given in Step 3).

Step 4. Call SUBROUTINE3(ON, $\frac{1}{6}$, $1 - b$, u), and $s :=$ (the return value).
Then, $z + w :=$ (# bins with one item given in Step 1 and one given in Step 4), and $v :=$ (# bins with exactly one item given in Step 4).

Step 5. Give ON $u + z + w$ items of size $1 - s$.

For an arbitrary online algorithm ALG and a positive integer `Length`, let ROUTINE1, 2, and 3 run and generate sequences of items σ_1, σ_2, and σ_3, respectively. What should be remarked

upon here is that σ_1 is a prefix of σ_2 and σ_2 is a prefix of σ_3. (This verifies the consistency of the variables $z(\geq 0)$ and $w(\geq 0)$ set in ROUTINE2 and 3.) Now we see what items are included in the longest sequence σ_3. According to the values t, b, and s determined through the execution of ROUTINE3, we classify all items into the following eight categories:

- $\mathtt{t}_-$-*items*, those which are of size in $(\frac{1}{10}, t]$ and given in Step 1,
- $\mathtt{t}_+$-*items*, those which are of size in $(t, \frac{1}{9})$ and given in Step 1,
- $(\mathtt{1}-\mathtt{t})$-*items*, those which are of size $(1-t)$ and given in Step 2,
- $\mathtt{b}_-$-*items*, those which are of size in $(\frac{4}{5}, b]$ and given in Step 3,
- $\mathtt{b}_+$-*items*, those which are of size in $(b, \frac{7}{8})$ and given in Step 3,
- $\mathtt{s}_-$-*items*, those which are of size in $(\frac{1}{6}, s]$ and given in Step 4,
- $\mathtt{s}_+$-*items*, those which are of size in $(s, 1-b)$ and given in Step 4, and
- $(\mathtt{1}-\mathtt{s})$-*items*, those which are of size $(1-s)$ and given in Step 5.

Lemma 1 clarifies the magnitude relation among items given in each subroutine. Together with the categorization above, we have the next fact:

Fact. *In the execution of* SUBROUTINE1 *(2, 3, respectively) called by* ROUTINE3*, an item that is put into a bin at Case 2.1 is classified as a* $\mathtt{t}_+$*-item (*$\mathtt{b}_+$*-item,* $\mathtt{s}_+$*-item, respectively), while one that is put into a bin at Case 2.2 is classified as a* $\mathtt{t}_-$*-item (*$\mathtt{b}_-$*-item,* $\mathtt{s}_-$*-item, respectively).*

The lemma below counts the numbers of non-empty bins of ALG and OPT. See Figure 1 for the packings. The proof will appear in the full version.

Lemma 2. *For ALG, OPT, and (x, y, u, z, w, v) determined by each of* ROUTINE1*, 2, and 3, it holds that:*

$$\begin{aligned}
\mathtt{Length} &= x + y, \\
C_{ALG}(\sigma_1) &= x + y, \\
C_{OPT}(\sigma_1) &= \frac{1}{2}x + \left\lceil \frac{1}{4}x + \frac{1}{2}y \right\rceil \leq \frac{3}{4}x + \frac{1}{2}y + \frac{1}{2}, \\
C_{ALG}(\sigma_2) &\geq \frac{3}{2}x + 2y + \frac{1}{2}(u - z), \\
C_{OPT}(\sigma_2) &= x + y + u, \\
C_{ALG}(\sigma_3) &\geq \frac{3}{2}x + y + 3u + v + 2z + 2w, \\
C_{OPT}(\sigma_3) &= x + y + 2u + z + w + \left\lceil \frac{1}{2}v \right\rceil \leq x + y + 2u + z + w + \frac{1}{2}v + \frac{1}{2}.
\end{aligned}$$

The next lemma is the heart of our analysis, which follows from Lemmas 4 and 5. The proof of Lemma 4 is omitted here.

Lemma 3. *For an arbitrary online algorithm ALG and any $\varepsilon > 0$, there exists a positive integer* $\mathtt{Length}$ *such that: Let* ROUTINE1*, 2, and 3 run with ALG and* $\mathtt{Length}$ *as parameters, and generate sequences of items σ_1, σ_2, and σ_3, respectively. Then, it follows that*

$$\max\left\{\frac{C_{ALG}(\sigma_1)}{C_{OPT}(\sigma_1)}, \frac{C_{ALG}(\sigma_2)}{C_{OPT}(\sigma_2)}, \frac{C_{ALG}(\sigma_3)}{C_{OPT}(\sigma_3)}\right\} \geq \overline{r} - \varepsilon, \tag{1}$$

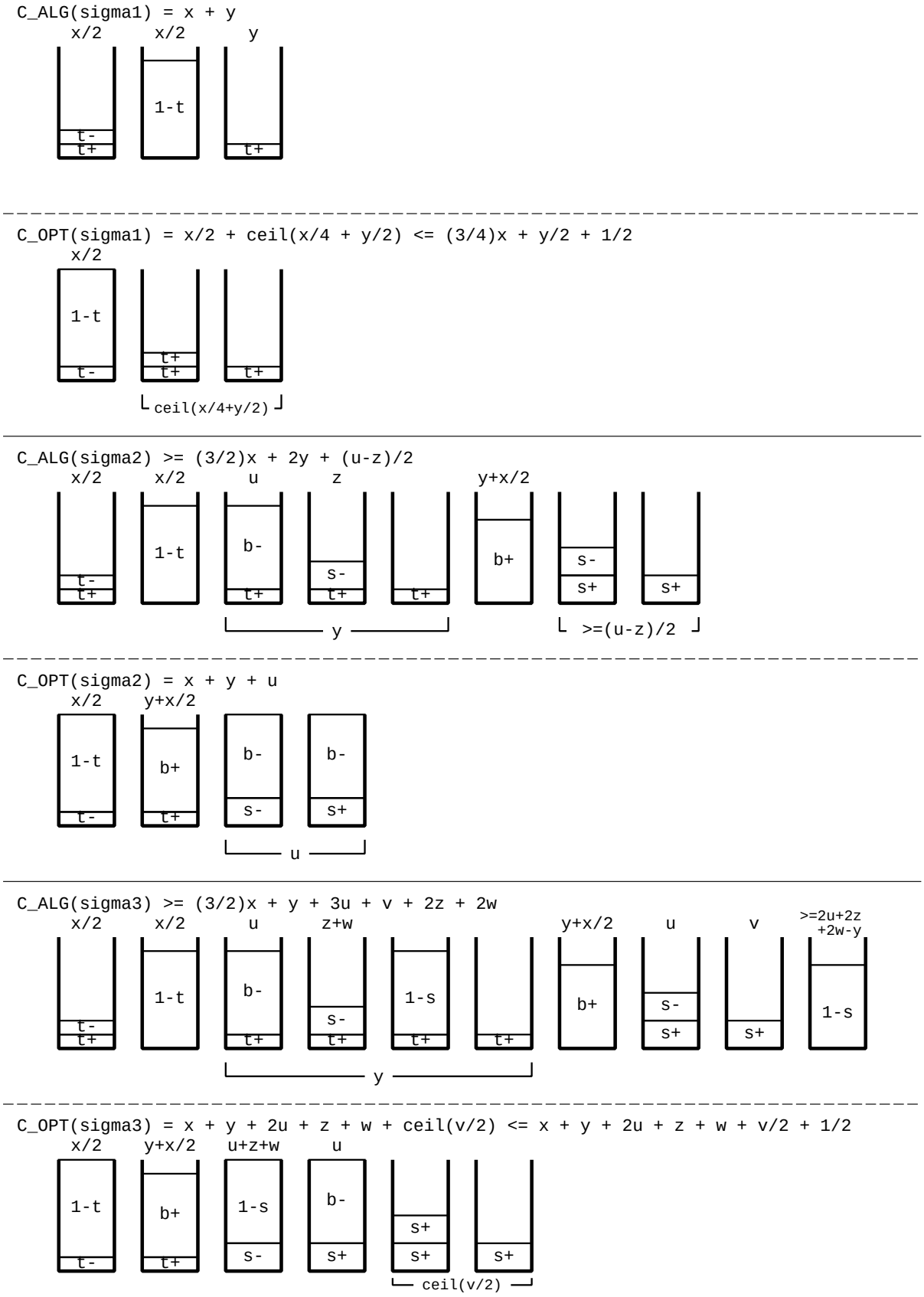

Figure 1: Packings by an arbitrary online algorithm and an optimal offline algorithm for the input sequences generated by ROUTINE1, 2, and 3. The numbers above (or below) bins indicate the number of bins belonging to that type.

where $\overline{r}(\approx 1.42764)$ *is the root of the cubic equation* $2r^3 - 17r^2 + 30r - 14 = 0$ *which lies between* $\frac{4}{3}$ *and* $\frac{3}{2}$.

Lemma 4. *For any* $\varepsilon > 0$*, there exists a positive integer* `Length` *such that for any nonnegative integers* $x, y, u, z, w,$ *and* v *with* $x + y =$ `Length`*,*

$$\frac{x+y}{\frac{3}{4}x + \frac{1}{2}y} - \frac{x+y}{\frac{3}{4}x + \frac{1}{2}y + \frac{1}{2}} \leq \varepsilon, \tag{2}$$

$$\frac{\frac{3}{2}x + y + 3u + 2z + 2w + v}{x + y + 2u + z + w + \frac{1}{2}v} - \frac{\frac{3}{2}x + y + 3u + 2z + 2w + v}{x + y + 2u + z + w + \frac{1}{2}v + \frac{1}{2}} \leq \varepsilon. \tag{3}$$

Lemma 5. *For any nonnegative integers* $x, y, u, z, w,$ *and* v *with* $x + y > 0$*,*

$$\max\left\{\frac{x+y}{\frac{3}{4}x + \frac{1}{2}y}, \frac{\frac{3}{2}x + 2y + \frac{1}{2}(u-z)}{x+y+u}, \frac{\frac{3}{2}x + y + 3u + v + 2z + 2w}{x + y + 2u + z + w + \frac{1}{2}v}\right\} \geq \overline{r}. \tag{4}$$

Proof. The proof is done by contradiction. Assume the lemma to be false. Then, all of the operands of the max operation in (4) can fall below $\overline{r}$ in the same time. That is to say, there exists a tuple of nonnegative integers (x, y, u, z, w, v) with $x + y > 0$ such that

$$f_1 := x + y - \overline{r}\left(\frac{3}{4}x + \frac{1}{2}y\right) < 0, \tag{5}$$

$$f_2 := \frac{3}{2}x + 2y + \frac{1}{2}(u - z) - \overline{r}\,(x + y + u) < 0, \tag{6}$$

$$f_3 := \frac{3}{2}x + y + 3u + v + 2z + 2w - \overline{r}\left(x + y + 2u + z + w + \frac{1}{2}v\right) < 0. \tag{7}$$

In what follows we show that there is no such (x, y, u, z, w, v). Specifically, we derive an inequality that does not contain either x, y, or u from the inequalities (5), (6), and (7). We then claim that there do not exist z, w, and v which satisfy the derived inequality.

Recall $\frac{4}{3} < \overline{r} < \frac{3}{2}$. Noting that $3 - 2\overline{r}$ and $2\overline{r} - 1$ are both positive, we have an inequality without u from (6) and (7).

$$\begin{aligned} f_4 :=& 4(3 - 2\overline{r})f_2 + 2(2\overline{r} - 1)f_3 \\ =& \left(-2\overline{r}^2 + 5\overline{r} - 2\right)v + \left(-4\overline{r}^2 + 10\overline{r} - 4\right)w + \left(4\overline{r}^2 - 16\overline{r} + 15\right)x \\ &+ \left(4\overline{r}^2 - 22\overline{r} + 22\right)y + \left(-4\overline{r}^2 + 14\overline{r} - 10\right)z \\ <& 0. \end{aligned}$$

(Please see that the elimination is done so that the resulting inequality sign makes sense.) Next, let us eliminate x. The coefficient of x in the above inequality $4\overline{r}^2 - 16\overline{r} + 15 = (2\overline{r} - 5)(2\overline{r} - 3)$ is confirmed to be positive. Together with positivity of $3\overline{r} - 4$, we eliminate x using (5).

$$\begin{aligned} f_5 :=& 4(4\overline{r}^2 - 16\overline{r} + 15)f_1 + (3\overline{r} - 4)\,f_4 \\ =& \left(-6\overline{r}^3 + 23\overline{r}^2 - 26\overline{r} + 8\right)(v + 2w) + \\ & 2\left(2\overline{r}^3 - 17\overline{r}^2 + 30\overline{r} - 14\right)y + 2\left(-6\overline{r}^3 + 29\overline{r}^2 - 43\overline{r} + 20\right)z \\ =& \left(-6\overline{r}^3 + 23\overline{r}^2 - 26\overline{r} + 8\right)(v + 2w) + 2\left(-6\overline{r}^3 + 29\overline{r}^2 - 43\overline{r} + 20\right)z \\ <& 0. \end{aligned}$$

The reason why y has gone is because $\overline{r}$ is a root of $2r^3 - 17r^2 + 30r - 14 = 0$. $\frac{4}{3} < \overline{r} < \frac{3}{2}$ leads to that both $(-6\overline{r}^3 + 23\overline{r}^2 - 26\overline{r} + 8)$ and $(-6\overline{r}^3 + 29\overline{r}^2 - 43\overline{r} + 20)$ are positive. Therefore, for fulfilling $f_5 < 0$, either z, w, or v should be negative. This contradicts the assumption that z, w, and v are all nonnegative. □

Our new lower bound is obtained almost as a corollary from Lemma 3.

Theorem 1. *Any online algorithm for the online 2-item bin packing problem has an asymptotic competitive ratio of at least $\overline{r}$, where $\overline{r}(\approx 1.42764)$ is the root of the cubic equation $2r^3 - 17r^2 + 30r - 14 = 0$ which lies between $\frac{4}{3}$ and $\frac{3}{2}$.*

3 Lower Bounds for $k \geq 4$

We propose an approach for deriving a lower bound of the online k-item bin packing problem for each $k \geq 4$, expanding the method of van Vliet [vV92] for the problem without a cardinality constraint. His method was to solve a linear program in which variables represent the packings by an arbitrary online algorithm given some patterns of input sequences. We illustrate how to embed a cardinality constraint into the linear program.

Some existing lower bounds for the problem without a cardinality constraint, such as [Yao80, vV92, BBG12], can be interpreted as lower bounds for the online k-item bin packing for some ranges of k; if the possible item size is restricted to be at least s, then the problem can be seen as the online k-item bin packing for $k \geq \lfloor \frac{1}{s} \rfloor$, since there is no chance that more than $\lfloor \frac{1}{s} \rfloor$ items are packed together. Such results include: A lower bound of $\frac{4}{3}$ for $4 \leq k \leq 5$ [vV92], $\frac{3}{2}$ for $6 \leq k \leq 41$ [Yao80], and $\frac{217}{141}$ for $42 \leq k$ [vV92]. See Table 1 in Section 1. Note that although the paper [vV92] does not provide the value of $\frac{4}{3}$ explicitly, it is given just by slightly changing the settings of his method. In the derivation of these results, it is not assumed that an algorithm packs items so that the cardinality constraint is kept. After the reformulation of a linear program, we set $k < \lfloor \frac{1}{s} \rfloor$ and try to obtain a better lower bound.

We first give our new formulation with the cardinality constraint. We are given a tuple of item sizes $(s_1, \ldots, s_l)$ with $\sum_{i=1}^{l} s_i \leq 1$. Set N a positive integer divisible by k and $\lfloor \frac{1}{\sum_{h=i}^{l} s_h} \rfloor$ for all $1 \leq i \leq l$. Let L_i be a sequence of N items of size s_i for each $1 \leq i \leq l$. The adversary issues any of $L_l \cdots L_i$ $(1 \leq i \leq l)$.

We denote by a vector $(t_1, \ldots, t_l)^T$ a packing of a bin that consists of t_i items of size s_i for $1 \leq i \leq l$. Any packing has to satisfy the following constraints: (i) the capacity constraint $\sum_{i=1}^{l} t_i s_i \leq 1$, (ii) the cardinality constraint $\sum_{i=1}^{l} t_i \leq k$, and (iii) the constraint that only non-empty bins are taken into account $\sum_{i=1}^{l} t_i > 0$.

Sort all feasible packings in a lexicographical order with an entry of an item size with a larger index having a bigger priority. For example, for $(s_1, s_2, s_3) = (\frac{1}{2} + \varepsilon, \frac{1}{3} + \varepsilon, \frac{1}{7} + \varepsilon)$ and $k = 4$, we have $(1,0,0)^T, (0,1,0)^T, (1,1,0)^T, \ldots, (1,0,3)^T$, $(0,1,3)^T, (0,0,4)^T$. Denote by $t_{i,j}$ the i-th entry of the j-th packing in the sorted list. The set of $t_{i,j}$'s is regarded as a matrix. For the above example,

$$(t_{i,j}) = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 2 & 0 & 0 & 1 & 1 & 2 & 0 & 0 & 1 & 2 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 3 & 3 & 3 & 4 \end{pmatrix}.$$

Let m be the number of feasible packings, which is 17 for the example.

Fix an online algorithm ALG arbitrarily. Suppose that given the input sequence $L_l \cdots L_1$, ALG creates n_j bins with the j-th packing (i.e., $(t_{1,j}, \ldots, t_{l,j})^T$). Define p_i as the index of the first column that has a non-zero entry in the i-th row of the matrix $(t_{i,j})$. Then it holds that

for $i \geq 2$ a packing before the p_i-th is one that skips all of items of size $s_l, \ldots, s_i$ and begins packing from s_{i-1}, and that $p_1 = 1$. For the above example, $p_1 = 1$, $p_2 = 2$, and $p_3 = 5$. Thus, we can describe the total number of non-empty bins of ALG for $L_l \cdots L_i$ $(1 \leq i \leq l)$ as

$$C_{ALG}(L_l \cdots L_i) = \sum_{j=p_i}^{m} n_j. \tag{8}$$

The total number of non-empty bins of an optimal offline algorithm OPT is bounded by a simple but nontrivial lemma. We will provide the proof in the full version.

Lemma 6. *For* $1 \leq i \leq l$, $C_{OPT}(L_l \cdots L_i) \leq \max\{\frac{N}{\lfloor \frac{1}{\sum_{h=i}^{l} s_h} \rfloor}, \frac{N(l-i+1)}{k}\}$.

As a matter of course, the whole set of bins created by ALG for $L_l \cdots L_1$ contains N items of size s_i for each $1 \leq i \leq l$. This fact is described as $\sum_{j=1}^{m} t_{i,j} n_j = N$ for all $1 \leq i \leq l$. Note that as long as this equation holds, the packings for $L_l \cdots L_i$ $(1 \leq i \leq l-1)$ are consistent as well. For later formulation, we rewrite this as

$$\sum_{j=1}^{m} \frac{t_{i,j} n_j}{N} - 1 = 0, \quad 1 \leq i \leq l. \tag{9}$$

The asymptotic competitive ratio R_{ALG} is asymptotically lower-bounded by R such that

$$\frac{C_{ALG}(L_l \cdots L_i)}{C_{OPT}(L_l \cdots L_i)} - R \leq 0, \quad 1 \leq i \leq l. \tag{10}$$

A sufficient condition for (10) with slack variables $(u_1, \ldots, u_l)$ is

$$\min\Big\{ \Big\lfloor \frac{1}{\sum_{h=i}^{l} s_h} \Big\rfloor, \frac{k}{l-i+1} \Big\} \sum_{j=p_i}^{m} \frac{n_j}{N} + u_i - R = 0, \quad 1 \leq i \leq l \tag{11}$$

for some $u_i \geq 0$ $(1 \leq i \leq l)$. The derivation follows from (8) and $\frac{N}{C_{OPT}(L_l \cdots L_i)} \geq \min\{\lfloor \frac{1}{\sum_{h=i}^{l} s_h} \rfloor, \frac{k}{l-i+1}\}$ obtained from Lemma 6.

The problem of finding the minimum R that satisfies (9) and (11) is formulated as a mathematical program $(\mathcal{P}_N)$ with a $2l \times (m+l+1)$-matrix $A = (a_{i,j})$ and vectors $\boldsymbol{x}$, $\boldsymbol{b}$, and $\boldsymbol{c}$ as below.

$$a_{i,j} = \begin{cases} t_{i,j}, & 1 \leq i \leq l, 1 \leq j \leq m; \\ 0, & 1 \leq i \leq l, m+1 \leq j \leq m+l+1; \\ 0, & l+1 \leq i \leq 2l, 1 \leq j \leq p_{i-l} - 1; \\ \min\{\lfloor \frac{1}{\sum_{h=i}^{l} s_h} \rfloor, \frac{k}{l-i+1}\}, & l+1 \leq i \leq 2l, p_{i-l} \leq j \leq m; \\ \delta_{i-l, j-m}, & l+1 \leq i \leq 2l, m+1 \leq j \leq m+l; \\ -1, & l+1 \leq i \leq 2l, j = m+l+1. \end{cases}$$

$$\boldsymbol{x} = \Big(\frac{n_1}{N}, \ldots, \frac{n_m}{N}, u_1, \ldots, u_l, R\Big)^T, \boldsymbol{b} = (\overbrace{1, \ldots, 1}^{l}, \overbrace{0, \ldots, 0}^{l})^T, \boldsymbol{c} = (\overbrace{0, \ldots, 0}^{m+l}, 1)^T$$

$$\begin{aligned} (\mathcal{P}_N) \quad & \text{minimize } \boldsymbol{c}^T \boldsymbol{x} \\ & \text{subject to } A\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \geq 0, \boldsymbol{x} = \Big(\frac{n_1}{N}, \ldots, \frac{n_m}{N}, u_1, \ldots, u_l, R\Big) \\ & (n_1, \ldots, n_m) \in \mathbb{Z}^m, (u_1, \ldots, u_l, R) \in \mathbb{R}^{l+1} \end{aligned}$$

Table 2: Our new lower bounds for each $4 \leq k \leq 45$. Bold font indicates improvement.

k	lower bound	k	lower bound	k	lower bound
4	$\mathbf{\frac{3}{2}}(= \mathbf{1.5})$	18	$\mathbf{\frac{93}{61}}(\approx \mathbf{1.52459})$	32	$\mathbf{\frac{496}{323}}(\approx \mathbf{1.53560})$
5	$\mathbf{\frac{25}{17}}(\approx \mathbf{1.47058})$	19	$\mathbf{\frac{171}{112}}(\approx \mathbf{1.52678})$	33	$\mathbf{\frac{341}{222}}(\approx \mathbf{1.53603})$
6	$\frac{3}{2}(= 1.5)$	20	$\mathbf{\frac{315}{206}}(\approx \mathbf{1.52912})$	34	$\mathbf{\frac{527}{343}}(\approx \mathbf{1.53644})$
7	$\frac{3}{2}(= 1.5)$	21	$\mathbf{\frac{26}{17}}(\approx \mathbf{1.52941})$	35	$\mathbf{\frac{1085}{706}}(\approx \mathbf{1.53682})$
8	$\frac{3}{2}(= 1.5)$	22	$\mathbf{\frac{341}{223}}(\approx \mathbf{1.52914})$	36	$\mathbf{\frac{186}{121}}(\approx \mathbf{1.53719})$
9	$\frac{3}{2}(= 1.5)$	23	$\mathbf{\frac{713}{466}}(\approx \mathbf{1.53004})$	37	$\mathbf{\frac{1147}{746}}(\approx \mathbf{1.53753})$
10	$\mathbf{\frac{80}{53}}(\approx \mathbf{1.50943})$	24	$\mathbf{\frac{124}{81}}(\approx \mathbf{1.53086})$	38	$\mathbf{\frac{589}{383}}(\approx \mathbf{1.53785})$
11	$\mathbf{\frac{44}{29}}(\approx \mathbf{1.51724})$	25	$\mathbf{\frac{775}{506}}(\approx \mathbf{1.53162})$	39	$\mathbf{\frac{403}{262}}(\approx \mathbf{1.53816})$
12	$\mathbf{\frac{66}{43}}(\approx \mathbf{1.53488})$	26	$\mathbf{\frac{403}{263}}(\approx \mathbf{1.53231})$	40	$\mathbf{\frac{20}{13}}(\approx \mathbf{1.53846})$
13	$\mathbf{\frac{26}{17}}(\approx \mathbf{1.52941})$	27	$\mathbf{\frac{279}{182}}(\approx \mathbf{1.53296})$	41	$\mathbf{\frac{1271}{826}}(\approx \mathbf{1.53874})$
14	$\mathbf{\frac{441}{289}}(\approx \mathbf{1.52595})$	28	$\mathbf{\frac{434}{283}}(\approx \mathbf{1.53356})$	42	$\frac{1519}{993}(\approx 1.52970)$
15	$\mathbf{\frac{315}{206}}(\approx \mathbf{1.52912})$	29	$\mathbf{\frac{899}{586}}(\approx \mathbf{1.53412})$	43	$\frac{9331}{6098}(\approx 1.53017)$
16	$\mathbf{\frac{624}{409}}(\approx \mathbf{1.52567})$	30	$\mathbf{\frac{155}{101}}(\approx \mathbf{1.53465})$	44	$\frac{4774}{3119}(\approx 1.53061)$
17	$\mathbf{\frac{527}{346}}(\approx \mathbf{1.52312})$	31	$\mathbf{\frac{961}{626}}(\approx \mathbf{1.53514})$	45	$\frac{3255}{2126}(\approx 1.53104)$

Here $\delta_{\cdot,\cdot}$ is Kronecker delta (if $i = j$, $\delta_{i,j} = 1$; otherwise, $\delta_{i,j} = 0$).

In $A\boldsymbol{x} = \boldsymbol{b}$, the first l rows correspond to (9), while the $(l+1)$-th to $2l$-th rows correspond to (11). $\delta_{i-l,j-m}$ lets the slack variable u_i appear in the equation of the $(l+i)$-th row. The objective function is $\boldsymbol{c}^T\boldsymbol{x} = R$. Note that A, $\boldsymbol{b}$, and $\boldsymbol{c}$ are independent of the choice of N.

Apparently, the optimal value of the following linear program $(\mathcal{P})$ is a lower bound on the optimal value of $(\mathcal{P}_N)$.

$$(\mathcal{P}) \quad \begin{aligned} &\text{minimize } \boldsymbol{c}^T\boldsymbol{x} \\ &\text{subject to } A\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \geq 0, \boldsymbol{x} \in \mathbb{R}^{m+l+1} \end{aligned}$$

The next theorem provides a lower bound for each $4 \leq k \leq 45$. The reason why we do not mention $k \geq 46$ is simply because of space limitation. Note that as long as the computer power is available, one can calculate a lower bound for arbitrary k using our method. The proof is left to the full version.

Theorem 2. *For each $4 \leq k \leq 45$, any online algorithm for the online k-item bin packing problem has an asymptotic competitive ratio of at least the value in Table 2.*

One can see that the new lower bounds for some k, such as $k = 5$ or 13, are lower than that for smaller k. We believe, however, that the matching upper and lower bound increases with respect to k and approaches that for the problem without a cardinality constraint. The anomaly suggests a limit of our method for some values of k. It is an interesting open problem to construct a better scheme for a lower bound for arbitrary k.

References

[BBG12] J. Balogh, J. Békési, and G. Galambos. New lower bounds for certain classes of bin packing algorithms. *Theor. Comput. Sci.*, 440-441:1–13, 2012.

[BCKK04] L. Babel, B. Chen, H. Kellerer, and V. Kotov. Algorithms for on-line bin-packing problems with cardinality constraints. *Discrete Applied Mathematics*, 143(1-3):238–251, 2004.

[BE98] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[CKP03] A. Caprara, H. Kellerer, and U. Pferschy. Approximation schemes for ordered vector packing problems. *Naval Research Logistics*, 50(1):58–69, 2003.

[EL10] L. Epstein and A. Levin. AFPTAS results for common variants of bin packing: A new method for handling the small items. *SIAM Journal on Optimization*, 20(6):3121–3145, 2010.

[Eps06] L. Epstein. Online bin packing with cardinality constraints. *SIAM J. Discrete Math.*, 20(4):1015–1030, 2006.

[KP99] H. Kellerer and U. Pferschy. Cardinality constrained bin-packing problems. *Annals of Operations Research*, 92(0):335–348, 1999.

[KSS75] K. L. Krause, V. Y. Shen, and H. D. Schwetman. Analysis of several task-scheduling algorithms for a model of multiprogramming computer systems. *J. ACM*, 22(4):522–550, 1975.

[KSS77] K. L. Krause, V. Y. Shen, and H. D. Schwetman. Errata: "Analysis of several task-scheduling algorithms for a model of multiprogramming computer systems". *J. ACM*, 24(3):527, 1977.

[RBLL89] P. V. Ramanan, D. J. Brown, C. C. Lee, and D. T. Lee. On-line bin packing in linear time. *J. Algorithms*, 10(3):305–326, 1989.

[Sei02] S. S. Seiden. On the online bin packing problem. *J. ACM*, 49(5):640–671, 2002.

[ST85] D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Commun. ACM*, 28(2):202–208, 1985.

[vV92] A. van Vliet. An improved lower bound for on-line bin packing algorithms. *Inf. Process. Lett.*, 43(5):277–284, 1992.

[Yao80] A. C. Yao. New algorithms for bin packing. *J. ACM*, 27(2):207–227, 1980.