Lower Bound for the On-line Bin Packing Problem with Restricted Repacking *

János Balogh+, József Békési+, Gábor Galambos+
Gerhard Reinelt‡

+Department of Computer Science,
University of Szeged,
Faculty of Juhász Gyula Teacher Training College,
6701 Szeged, POB. 396.
E-mail:{balogh,bekesi,galambos}@jgytf.u-szeged.hu

‡Institute for Computer Science
University of Heidelberg
E-mail:gerhard.reinelt@informatik.uni-heidelberg.de

Abstract

In 1996 Ivkovič and Lloyd [11] gave the lower bound $\frac{4}{3}$ on the asymptotic worst-case ratio for so-called fully dynamic bin packing algorithms, where the number of repackable items in each step is restricted by a constant. In this paper we improve this result to about 1.3871. We present our proof for a semi-on-line case of the classical bin packing, but it works for fully dynamic bin packing as well. We prove the lower bound by analyzing and solving a specific optimization problem. The bound can be expressed exactly using the Lambert W function.

1 Introduction

The classical one-dimensional bin packing problem is among the most frequently studied combinatorial optimization problems. In its traditional definition a list $L = \{x_1, x_2, \ldots, x_n\}$ of elements (also called items) with sizes in the interval $(0, 1]$ and an infinite list of unit capacity bins are given. Each element x_i from

*The research was supported by the Hungarian National Research Fund (projects T 048377 and T 046822) and by the MÖB-DAAD Hungarian-German Researcher Exchange Program (project No. 21).
A preliminary version of the paper was presented at the 29th Hungarian Conference of Mathematics, Physics and Computer Science Teachers and appeared in electronical form in the proceedings of the conference published by the organizers.

the list L has to be assigned to a unique bin such that the sum of the sizes of the elements in a bin does not exceed the bin capacity. The size of an element will be also denoted by x_i. The *bin packing problem* consists of packing the items to the bins in such a way that as few bins as possible are used.

It is well-known that finding an optimal packing is *NP-hard* [8]. Consequently, a large number of papers have been published which look for polynomial time algorithms that find feasible solutions with an acceptable approximation quality.

For measuring the efficiency of algorithms there are two general methods: the investigation of the worst-case behavior or – assuming some probability distribution of the elements – a probabilistic analysis. In this paper we will concentrate on the asymptotic worst-case ratio of an algorithm. For a given list L, denoted by $A(L)$ and $OPT(L)$ the number of bins used by algorithm A and the number of bins used in an optimal packing, respectively. Then the *asymptotic worst–case ratio (AWR)* of algorithm A is

$$R_A := \lim_{k\to\infty} \sup \left\{ \max_L \left\{ \frac{A(L)}{k} \,\middle|\, OPT(L) = k \right\} \right\}.$$

If an algorithm has an AWR of R_A, we also say that it is R_A*-competitive.*

In the last decades many different types of algorithms have been proposed and also several variants of the classical problem have been introduced and studied. Since the terminology used for classifying algorithms or problems is not unique, we start the paper with a proposal for a new classification based on a different view of the problem. To this end, we distinguish two contributors: the *scheduler* who provides information on the problem data and the *loader* who packs the items. The actual problem then depends on how the scheduler gives the data and on which strategies the loader uses for packing.

We suppose that the scheduler reveals information about the items at subsequent discrete time steps. The information consists of messages that certain elements (with respective sizes) arrive and/or that certain elements can be deleted. The total number of such time steps is at most n because information about several items can be given in the same step. If the element x_i appears in step j to be added to the list, then we say that its arrival time a_i is j. In some step k, the scheduler may also inform about the fact that the item x_i can be deleted. We say that the deletion time d_i of x_i is k. From this time the item may be deleted from the bin into which it has been packed earlier. Of course, $a_i < d_i$, for $1 \le i \le n$. By setting $d_i = \infty$ we indicate that no deletion time for item x_i has been given.

At every time step the loader decides on its actions for packing the items. If some elements are packed and some others are deleted, we assume that first the deletion operations are performed.

In short, the scheduler decides on how the problem data is given, and the loader finds a feasible packing of all items depending on arrival and deletion times. Special strategies now lead to different variants of bin packing.

If $a_i = 1$ and $d_i = \infty$ for every i, $1 \le i \le n$, then the loader immediately receives complete knowledge about the list. In this case the scheduler produces an *off-line problem* and the loader applies an *off-line algorithm*.

If $a_i = i$ and $d_i = \infty$ and the loader packs the items one by one in each step without knowing anything about subsequent elements (neither the sizes nor the number of the elements), and the packed items may not be repacked anymore during the algorithm, then we speak about an *on-line problem* and an *on-line algorithm*.

In the case of *semi-online (*SOL*) algorithms* the scheduler specifies the problem on-line as above, but the loader is allowed to apply at least one of the following operations: repacking of some items, lookahead into several next elements or some kind of preordering. Semi-online algorithms allowing only a restricted number of elements to be repacked in each step are called *c-repacking semi-online algorithms*.

If the scheduler also allows the deletion of elements then we speak about a *dynamic bin-packing problem* (*DBP*), and if the loader exploits this additional information for deleting items then his algorithm is a *dynamic bin-packing algorithm*. In this case $A(L)$ is considered as the maximum number of used bins during the packing. An off-line dynamic bin packing problem basically would be a special type of scheduling problem. Therefore, in the context of bin packing, only *on-line and semi-online dynamic problems* are investigated.

If, in the semi-online case, only repacking is allowed for the loader and the scheduler may specify some elements to be deleted, then Ivkovič and Lloyd in [12] use the terms *fully-dynamic bin-packing problem* (*FDBP*) and *fully-dynamic bin-packing algorithm*. Similarly to the definition of c-repacking semi-online problems, the c-repacking FDBP can be defined.

For off-line bin packing algorithms the best results were achieved in [5] and [13], where the authors proved that for any $\varepsilon > 0$, there are algorithms, which can solve the bin packing problem in linear time, while their AWR is $1 + \varepsilon$. Recently, the best on-line algorithm was given by Seiden in [15] and its AWR is 1.58889. The best known lower bound 1.5401 is due to van Vliet [16].

SOL algorithms were investigated in [9],[10], and [6]. The first and the second one use different types of lookaheads, and the last one allows repacking. Gambosi et al. [7] also analyzed certain SOL algorithms. In their algorithms repacking was allowed in a special way: those elements which were "large enough" were repacked one by one, while the "small" elements were moved together in a bundle between the bins. In this sense these algorithms can repack even $O(n)$ elements in one step. They analyzed two algorithms. The faster one has linear time and is $\frac{3}{2}$-competitive, the other one runs in $O(n \log n)$ time and is $\frac{4}{3}$-competitive. Recently, except for trivial results no good lower bound is known for the efficiency of SOL algorithms.

Various dynamic approximation algorithms were investigated in [3].

Ivkovič and Lloyd constructed an approximation algorithm to solve the FDBP (see [12]). Similar to the technique used in [7] they also repacked the small elements using a "bundle technique". The competitive ratio of their algorithm is $\frac{5}{4}$, and requires $\Theta(\log n)$ time per step, i.e. its running time is $\Theta(n \log n)$. Investigating the lower bounds they proved that there is no FDBP algorithm with deletions and where only a constant number of elements may be repacked with better competitive ratio than $\frac{4}{3}$ [11] (i.e. for the c-repacking FDBP for any c). Using a similar construction in the proof this bound can be applied to SOL algorithms as well. This fact was also mentioned by Csirik and Woeginger in [4].

In this paper we improve the lower bound to 1.3871 for the c-repacking SOL algorithms, but it follows from the construction that this lower bound is also valid for c-repacking fully-dynamic bin-packing algorithms. The last fact is coming from the following observation. While we try to construct lower bounds either for the c-repacking SOL problem or the c-repacking FDBP we can realize that the two constructions differ slightly from each other: to allow some deletions will not spoil the c-repacking SOL lower-bound construction for the c-repacking FDBP case.

During our analysis we will use different instruments: LP-techniques will be combined with results from linear algebra, and finally we will solve a non-linear optimization problem.

2 Construction of the linear program

For $k \geq 1$, $n \geq 1$, let $x_1, x_2, \ldots, x_k$ $\left(\frac{1}{2} \leq x_1 < x_2 < \ldots < x_k < 1\right)$ be fixed real numbers and $c \geq 1$ be an arbitrary integer. Define $y_i = 1 - x_i\,(i = 1, \ldots, k)$ and $y_{k+1} = 0$. Furthermore let $\varepsilon < \min\limits_{j=1,\ldots,k} \{y_j - y_{j+1}\}$ be an arbitrary positive number and $\varepsilon_j := \frac{\varepsilon}{\lceil \frac{n}{2y_j} \rceil}$.

Denote the first list by L_0. Define L_0 as a list of N items of size a, where $a < \frac{\varepsilon_k}{\lceil \frac{n}{y_k} \rceil c}$ and $N := \lfloor \frac{\frac{n}{2} - \varepsilon}{a} \rfloor$. We can see that L_0 contains very small elements with equal, suitably chosen size. The cumulative size of the small items converges to $\frac{n}{2}$ from below, if $n \to \infty$. Denote by L_i $(1 \leq i \leq k, k \geq 2)$ the lists containing big elements. Define L_i as a list of $\lceil \frac{n}{2y_i} \rceil$ items of size $x_i + \varepsilon_i$. This means that the sizes of the elements are equal in list L_i. Suppose we need to pack the elements of the lists $L_0, L_1, \ldots, L_k$ with an SOL algorithm which can repack only a constant number of elements. We call this kind of algorithms c-repacking algorithms. We analyze the behavior of an arbitrary c-repacking algorithm for the lists L_0 and $L_0 L_i$ $(1 \leq i \leq k)$. In our analysis $size(B)$ denotes the total size of items in bin B. We present the lower bound as a solution of a linear programming problem.

Lemma 1

$$\limsup_{n\to\infty} \frac{A\left(L_0\right)}{OPT\left(L_0\right)} \geq 1 + \sum_{j=1}^{k} 2z_j \left(\frac{1}{y_j} - 1\right).$$

Proof. Consider an arbitrary c-repacking algorithm A, that has to pack list L_0 first. It is clear that the above defined a is a very small number, and the total size of the repackable items in $\lceil \frac{n}{2y_j} \rceil$ $(j = 1, \ldots, k)$ steps is less than ε_k, which is the smallest among $\varepsilon_1, \ldots, \varepsilon_k$. We want to give a lower bound on $A\left(L_0\right)$.

Denote by $z_i n$ the cumulative size of the items that have been packed in y_i-type bins ($i = 1, \ldots, k$). Bin B is called y_i-type bin if $size(B) \in (y_{i+1}, y_i]$. It is easy to see that the total number of y_i-type bins is at least $\frac{z_i n}{y_i}$. The cumulative size of the items that have not been packed in any y_i-type bin is $Na - n\sum_{j=1}^{k} z_j$. So we get that

$$A(L_0) \geq \sum_{j=1}^{k} \frac{z_j n}{y_j} + Na - n\sum_{j=1}^{k} z_j = Na + n\sum_{j=1}^{k} z_j \left(\frac{1}{y_j} - 1\right).$$

Because of the definition of N we have $Na \geq \frac{n}{2} - \varepsilon - a$. This means that

$$A(L_0) \geq n\left(\frac{1}{2} + \sum_{j=1}^{k} z_j \left(\frac{1}{y_j} - 1\right)\right) - \varepsilon - a. \tag{1}$$

In the following we investigate the behavior of the optimal algorithm on L_0. In the optimal packing each bin B is loaded such that $size(B) > 1 - a$. So we get

$$OPT(L_0) \leq \frac{Na}{1-a} + 1 \leq \frac{\frac{n}{2} - \varepsilon}{1-a} + 1 \leq \frac{\frac{n}{2}}{1-a} + 1. \tag{2}$$

From (1) and (2) it follows that

$$\limsup_{n\to\infty} \frac{A(L_0)}{OPT(L_0)} \geq 1 + \sum_{j=1}^{k} 2z_j \left(\frac{1}{y_j} - 1\right). \tag{3}$$

■

Lemma 2 *For $i = 1, \ldots, k$:*

$$\limsup_{n\to\infty} \frac{A(L_0 L_i)}{OPT(L_0 L_i)} \geq 1 + y_i + 2y_i \left(\sum_{j=1}^{i-1} z_j \left(\frac{1}{y_j} - 1\right) - \sum_{j=i}^{k} z_j\right).$$

Proof. Consider now the packing of the concatenated list $L_0 L_i$ ($1 \leq i \leq k$) for fixed i. Let B be a bin which contains an element from L_i. Because the size of the big item is $x_i + \varepsilon_i$, the total size of the small elements in the bin is at most $y_i - \varepsilon_i$. This can also happen in such a way, that after L_0 has been packed, the bin contains more items, but during the processing of L_i, some of them have been repacked to other bins. Since the list L_i has exactly $\lceil \frac{n}{2y_i} \rceil$ elements, at most $c\lceil \frac{n}{2y_i} \rceil$ elements could have been repacked. The size of each small item is a, so from the definition of a and ε_i we get that A can repack small items with at most ε_i cumulative size. So we can say that after packing L_0 $size(B) \leq y_i$ must hold for B. From this it follows that the cumulative size of the small items packed together with an item from L_i is $\sum_{j=i}^{k} z_j n$. Other small items from L_0 can be packed in y_j-type bins, where $j = 1, \ldots, i-1$. The total size of the remaining elements of L_0 is $Na - n\sum_{j=1}^{k} z_j$. So we can estimate the number of bins used by A in the following way:

$$A\left(L_0 L_i\right) \geq \frac{n}{2y_i} + n\sum_{j=1}^{i-1}\frac{z_j}{y_j} + Na - n\sum_{j=1}^{k} z_j, \quad (i = 1, \ldots, k)\,. \tag{4}$$

For the optimal packing of $L_0 L_i$ $(1 \leq i \leq k)$ we get that

$$OPT\left(L_0 L_i\right) \leq \left\lceil \frac{n}{2y_i} \right\rceil + S_i \leq \frac{n}{2y_i} + S_i + 1, \tag{5}$$

where S_i is the total number of bins, which contain items only from L_0 in an optimal packing. A bin containing an L_i-element will be denoted by B_i. We obtain for the cumulative size $small(B_i)$ of the small elements in B_i that

$$\begin{aligned} small(B_i) &\geq \left\lceil \frac{n}{2y_i} \right\rceil (y_i - \varepsilon_i - a) \geq \frac{n}{2} - \left\lceil \frac{n}{2y_i} \right\rceil \varepsilon_i - \left\lceil \frac{n}{2y_i} \right\rceil a \\ &\geq Na - \left\lceil \frac{n}{2y_i} \right\rceil a \geq Na - \varepsilon, \end{aligned}$$

because $a < \frac{\varepsilon}{\left\lceil \frac{n}{2y_i} \right\rceil}$. So

$$S_i \leq \frac{\varepsilon}{a} \leq \frac{1}{n^2} \leq 1. \tag{6}$$

From (5) and (6)

$$OPT\left(L_0 L_i\right) \leq \frac{n}{2y_i} + 2. \tag{7}$$

Combining (4) and (7) we get that

$$\begin{aligned} \limsup_{n\to\infty} \frac{A\left(L_0 L_i\right)}{OPT\left(L_0 L_i\right)} &\geq 1 + y_i + 2y_i \left(\sum_{j=1}^{i-1} z_j \left(\frac{1}{y_j} - 1\right) - \sum_{j=i}^{k} z_j \right), \\ &\quad (i = 1, \ldots, k)\,. \end{aligned} \tag{8}$$

■

Theorem 3 *The solution of the following linear programming problem is a lower bound for any semi-on-line c-repacking algorithm:*

$$\begin{aligned} &\min b \\ b &\geq 1 + y_i + 2y_i \left(\sum_{j=1}^{i-1} z_j \left(\frac{1}{y_j} - 1\right) - \sum_{j=i}^{k} z_j \right), \quad (i = 1, \ldots, k)\,, \\ b &\geq 1 + \sum_{j=1}^{k} 2z_j \left(\frac{1}{y_j} - 1\right), \\ z_i &\geq 0, (i = 1, \ldots, k)\,, \\ \sum_{j=1}^{k} z_j &< \frac{1}{2}. \end{aligned} \tag{9}$$

Proof. The definition of z_j $(j = 1, \dots, k)$ obviously implies that $\sum_{j=1}^{k} z_j < \frac{1}{2}$. The statement of the theorem then follows with Lemmas (1) and (2). ■

3 Solution of the linear program

To simplify our model we introduce a new variable z, which is the sum of all z_j's, i.e. $z = \sum_{j=1}^{k} z_j$. Using this substitution and some reordering (9) can be written in the following form:

$$\begin{aligned}
& \min \; b \\
-2y_i z + 2y_i \sum_{j=1}^{i-1} \frac{z_j}{y_j} - b &\leq -\left(1 + y_i\right), \quad \left(i = 1, \dots, k\right), \\
-2z + 2\sum_{j=1}^{k} \frac{z_j}{y_j} - b &\leq -1, \\
-z + \sum_{j=1}^{k} z_j &= 0 \\
z_i &\geq 0, \left(i = 1, \dots, k\right), \\
z &< \frac{1}{2}.
\end{aligned} \tag{10}$$

Instead of (10) we first solve a linear system of equations related to our problem. The number of equations is $k + 2$. The variables of the system are $z, z_1, \dots, z_k, b$ while $y_1, \dots, y_k$ are some fixed parameters, satisfying the conditions of Theorem 3.

$$\begin{aligned}
-2y_i z + 2y_i \sum_{j=1}^{i-1} \frac{z_j}{y_j} - b &= -\left(1 + y_i\right), \quad \left(i = 1, \dots, k\right), \\
-2z + 2\sum_{j=1}^{k} \frac{z_j}{y_j} - b &= -1, \\
-z + \sum_{j=1}^{k} z_j &= 0
\end{aligned} \tag{11}$$

Lemma 4 *The equation system (11) has a unique solution.*

Proof. Let M be the matrix of (11). Then

$$M = \begin{bmatrix} -2y_1 & 0 & \dots & 0 & 0 & -1 \\ -2y_2 & 2\frac{y_2}{y_1} & \dots & 0 & 0 & -1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ -2y_k & 2\frac{y_k}{y_1} & \dots & 2\frac{y_k}{y_{k-1}} & 0 & -1 \\ -2 & \frac{2}{y_1} & \dots & \frac{2}{y_{k-1}} & \frac{2}{y_k} & -1 \\ -1 & 1 & \dots & 1 & 1 & 0 \end{bmatrix} \tag{12}$$

Denote by $\det(M)$ the determinant of M. We prove that $\det(M) < 0$, and so from Cramer's Rule the statement of the lemma follows. Let us expand $\det(M)$ using the last column of M. So we obtain

$$\det(M) = \sum_{i=1}^{k+1} -1\,(-1)^{k+2+i}\det(M_i) = \sum_{i=1}^{k+1}(-1)^{k+1+i}\det(M_i), \tag{13}$$

where M_i is a $(k+1)\times(k+1)$ matrix, which is obtained from M by deleting its last column and i-th row. Expand $\det(M_i)$ by its last row. So

$$\det(M_i) = (-1)(-1)^{k+2}\det(M_{i1}) + \sum_{j=2}^{k+1}(-1)^{k+1+j}\det(M_{ij}). \tag{14}$$

where M_{ij} is a $k\times k$ matrix, which is obtained from M_i by deleting its last row and j-th column. Now we investigate, how M_{ij} looks like.

$$M_{ij} = \begin{bmatrix} -2y_1 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ -2y_2 & 2\frac{y_2}{y_1} & \dots & 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ -2y_{i-1} & 2\frac{y_{i-1}}{y_1} & \dots & 2\frac{y_{i-1}}{y_{j-2}} & 2\frac{y_{i-1}}{y_j} & \dots & 0 & 0 \\ -2y_{i+1} & 2\frac{y_{i+1}}{y_1} & \dots & 2\frac{y_{i+1}}{y_{j-2}} & 2\frac{y_{i+1}}{y_j} & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots \\ -2y_k & 2\frac{y_k}{y_1} & \dots & 2\frac{y_k}{y_{j-2}} & 2\frac{y_k}{y_j} & \dots & 2\frac{y_k}{y_{k-1}} & 0 \\ -2 & \frac{2}{y_1} & \dots & \frac{2}{y_{j-2}} & \frac{2}{y_j} & \dots & \frac{2}{y_{k-1}} & \frac{2}{y_k} \end{bmatrix}$$

Case A**:** Suppose that $j \neq i$ and $j \neq i+1$ and consider only the i-th and $i+1$-th columns of M_{ij}. The two columns are the following:

$$\begin{bmatrix} 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ 2\frac{y_{i+1}}{y_{i-1}} & 2\frac{y_{i+1}}{y_i} \\ \vdots & \vdots \\ 2\frac{y_k}{y_{i-1}} & 2\frac{y_k}{y_i} \\ \frac{2}{y_{i-1}} & \frac{2}{y_i} \end{bmatrix}$$

It is easy to see that these columns are not independent, so we obtain that $\det(M_{ij}) = 0$, if $j \neq i$ and $j \neq i+1$.

Case B: If $j = i$ then M_{ii} is a lower triangular matrix of the form:

$$M_{ii} = \begin{bmatrix} -2y_1 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ -2y_2 & 2\frac{y_2}{y_1} & \dots & 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ -2y_{i-1} & 2\frac{y_{i-1}}{y_1} & \dots & 2\frac{y_{i-1}}{y_{i-2}} & 0 & \dots & 0 & 0 \\ -2y_{i+1} & 2\frac{y_{i+1}}{y_1} & \dots & 2\frac{y_{i+1}}{y_{i-2}} & 2\frac{y_{i+1}}{y_i} & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots \\ -2y_k & 2\frac{y_k}{y_1} & \dots & 2\frac{y_k}{y_{i-2}} & 2\frac{y_k}{y_i} & \dots & 2\frac{y_k}{y_{k-1}} & 0 \\ -2 & \frac{2}{y_1} & \dots & \frac{2}{y_{i-2}} & \frac{2}{y_i} & \dots & \frac{2}{y_{k-1}} & \frac{2}{y_k} \end{bmatrix}$$

From this we get that $\det(M_{11}) = 2^k \frac{1}{y_1}$, $\det(M_{(k+1)(k+1)}) = -2^k y_k$ and that for $2 \leq i \leq k$, $\det(M_{ii}) = -2^k \frac{y_{i-1}}{y_i}$.

Case C: Similarly, if $j = i+1$ then $M_{i(i+1)}$ is also a lower triangular matrix:

$$M_{i(i+1)} = \begin{bmatrix} -2y_1 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ -2y_2 & 2\frac{y_2}{y_1} & \dots & 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ -2y_{i-1} & 2\frac{y_{i-1}}{y_1} & \dots & 2\frac{y_{i-1}}{y_{i-2}} & 0 & \dots & 0 & 0 \\ -2y_{i+1} & 2\frac{y_{i+1}}{y_1} & \dots & 2\frac{y_{i+1}}{y_{i-2}} & 2\frac{y_{i+1}}{y_{i-1}} & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots \\ -2y_k & 2\frac{y_k}{y_1} & \dots & 2\frac{y_k}{y_{i-2}} & 2\frac{y_k}{y_{i-1}} & \dots & 2\frac{y_k}{y_{k-1}} & 0 \\ -2 & \frac{2}{y_1} & \dots & \frac{2}{y_{i-2}} & \frac{2}{y_{i-1}} & \dots & \frac{2}{y_{k-1}} & \frac{2}{y_k} \end{bmatrix}$$

It is easy to see that $\det(M_{i(i+1)}) = -2^k$ if $i \neq 1$ and $\det(M_{12}) = 2^k$.
Using these results we get from (14) that

$$\det(M_1) = -2^k \left((-1)^{k+2} \frac{1}{y_1} + (-1)^{k+3} \right),$$

$$\det(M_i) = -2^k \left((-1)^{k+1+i} \frac{y_{i-1}}{y_i} + (-1)^{k+1+i+1} \right), i = 2, \ldots, k,$$

$$\det(M_{k+1}) = -2^k y_k.$$

So from (13) we obtain

$$\begin{aligned}
\det(M) = & -2^k \left((-1)^{2(k+1)} \frac{1}{y_1} + (-1)^{2(k+1)+1} \right) + \\
& + -2^k \left(\sum_{i=2}^{k} \left((-1)^{2(k+1+i)} \frac{y_{i-1}}{y_i} + (-1)^{2(k+1+i)+1} \right) + (-1)^{2(k+1)} y_k \right) \\
= & -2^k \left(\frac{1}{y_1} + \sum_{i=2}^{k} \frac{y_{i-1}}{y_i} + y_k - k \right).
\end{aligned}$$

Since $y_{i-1} > y_i$ $(i = 2, \ldots, k)$ and $\frac{1}{2} \geq y_1$ we get $\frac{1}{y_1} + \sum_{i=2}^{k} \frac{y_{i-1}}{y_i} > k$, and so $\det(M) < 0$. ■

Lemma 5 *The solution of equation system (11) satisfies the conditions of the linear program (10) and*

$$b = 1 + \frac{1 - y_k}{\frac{1}{y_1} + \sum\limits_{i=2}^{k} \frac{y_{i-1}}{y_i} + y_k - k}.$$

Proof. By substituting the right hand side of (11) into M we obtain the following matrix M_b:

$$M_b = \begin{bmatrix}
-2y_1 & 0 & \ldots & 0 & 0 & -1 - y_1 \\
-2y_2 & 2\frac{y_2}{y_1} & \ldots & 0 & 0 & -1 - y_2 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
-2y_k & 2\frac{y_k}{y_1} & \ldots & 2\frac{y_k}{y_{k-1}} & 0 & -1 - y_k \\
-2 & \frac{2}{y_1} & \ldots & \frac{2}{y_{k-1}} & \frac{2}{y_k} & -1 \\
-1 & 1 & \ldots & 1 & 1 & 0
\end{bmatrix},$$

and

$$\det(M_b) = \det(M) + \sum_{i=1}^{k} y_i \, (-1)^{k+1+i} \det(M_i) = \det(M) - 2^k \, (1 - y_k) \, .$$

Using again Cramer's Rule:

$$b = \frac{\det(M_b)}{\det(M)} = 1 + \frac{1-y_k}{\frac{1}{y_1} + \sum\limits_{i=2}^{k} \frac{y_{i-1}}{y_i} + y_k - k}.$$

By a similar analysis we can prove that the other conditions of (10) are also hold. As an example we present only the inequality

$$z = \frac{\det(M_z)}{\det(M)} = \frac{\frac{1}{2}\det(M) - 2^{k-1}\frac{y_k - 1}{y_1}}{\det(M)} = \frac{1}{2} + \frac{y_k - 1}{2y_1\left(\frac{1}{y_1} + \sum\limits_{i=2}^{k} \frac{y_{i-1}}{y_i} + y_k - k\right)} < \frac{1}{2}$$

where

$$M_z = \begin{bmatrix} -1-y_1 & 0 & \dots & 0 & 0 & -1-y_1 \\ -1-y_2 & 2\frac{y_2}{y_1} & \dots & 0 & 0 & -1-y_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ -1-y_k & 2\frac{y_k}{y_1} & \dots & 2\frac{y_k}{y_{k-1}} & 0 & -1-y_k \\ -1 & \frac{2}{y_1} & \dots & \frac{2}{y_{k-1}} & \frac{2}{y_k} & -1 \\ 0 & 1 & \dots & 1 & 1 & 0 \end{bmatrix}$$

The validity of the other inequalities can be shown easily. ■

Lemma 6 *The solution of equation system (11) is an optimal solution of the linear program (10).*

Proof. Consider the solution vector v of (11). Because of Lemma 5 it is a feasible solution of (10). Let v be a basis solution. The objective function contains only b, which has a positive coefficient, satisfying the optimum criterion. So we get the statement of the lemma. ■

4 Getting the Lower Bound

In the following part of the paper we deal with the optimal choice of the variables $y_1, y_2, \dots, y_k$. To choose them in an optimal way, we have to solve the following non-linear optimization problem:

$$\max\left\{1 + \frac{1-y_k}{\frac{1}{y_1} + \sum\limits_{i=2}^{k} \frac{y_{i-1}}{y_i} + y_k - k}\right\}$$

$$\frac{1}{2} \geq y_1 > \dots > y_k > 0. \tag{15}$$

It does not seem to be easy to solve (15) analytically for arbitrary fixed k, but it can be done numerically by global optimization tools. Details can be found in [1]. In this paper we analyze (15) for the case when $k \to \infty$.

Lemma 7 *The optimal solution of (15) converges to the maximum of the function*

$$f(x) := 1 + \frac{1-x}{x + 1 + \ln\left(\frac{1}{x}\right) - \ln(2)}$$

in the interval $\left(0, \frac{1}{2}\right]$ *if* $k \to \infty$.

Proof. We distinguish two cases:
Case 1: In this case we assume that $y_2 \le \frac{y_1}{2}$. Then $2 \le \frac{y_1}{y_2}$. Using $y_1 \le \frac{1}{2}, y_2 \le \frac{1}{4}, y_k \le y_2$, $4 \ge 2 + y_k$, and applying the inequality between the arithmetic and geometric means for the numbers $\frac{y_2}{y_3}, \frac{y_3}{y_4}, \dots, \frac{y_{k-1}}{y_k}, y_k$ in the denominator of (15), we obtain the upper approximation (see [14])

$$\begin{aligned} 1 + \frac{1 - y_k}{\frac{1}{y_1} + \sum\limits_{i=2}^{k} \frac{y_{i-1}}{y_i} + y_k - k} &\le 1 + \frac{1 - y_k}{4 + \sum\limits_{i=3}^{k} \frac{y_{i-1}}{y_i} + y_k - k} \\ &\le 1 + \frac{1 - y_k}{2 + y_k - k + (k-1)\sqrt[k-1]{y_k}}. \end{aligned}$$

Using that $\lim\limits_{k\to\infty} (k-1)\sqrt[k-1]{x} - k = \ln(x) - 1$ and $\ln(x) < -\frac{5}{4}$ if $x \le \frac{1}{4}$, we obtain that

$$\lim_{k\to\infty} 1 + \frac{1 - y_k}{\frac{1}{y_1} + \sum\limits_{i=2}^{k} \frac{y_{i-1}}{y_i} + y_k - k} \le 1 + \frac{1 - y_k}{y_k + 1 + \ln(y_k)} < 1.$$

Case 2: Suppose that $y_2 > \frac{y_1}{2}$. From this we get $2y_2(1 - 2y_1) \ge y_1(1 - 2y_1)$ and so

$$\frac{1}{y_1} + \frac{y_1}{y_2} \ge 2 + \frac{1}{2y_2}. \tag{16}$$

Using (16) and $y_1 \le \frac{1}{2}$ and then applying again the inequality between the arithmetic and geometric means for the numbers $\frac{1}{2y_2}, \frac{y_2}{y_3}, \dots, \frac{y_{k-1}}{y_k}$ in the denominator of (15), we obtain the upper approximation

$$\begin{aligned} 1 + \frac{1 - y_k}{\frac{1}{y_1} + \sum\limits_{i=2}^{k} \frac{y_{i-1}}{y_i} + y_k - k} &\le 1 + \frac{1 - y_k}{2 + \frac{1}{2y_2} + \sum\limits_{i=3}^{k} \frac{y_{i-1}}{y_i} + y_k - k} \\ &\le 1 + \frac{1 - y_k}{2 + y_k - k + (k-1)\sqrt[k-1]{\frac{1}{2y_k}}}. \end{aligned}$$

Since $\lim_{k\to\infty} (k-1)\sqrt[k-1]{\frac{1}{2x}} - k = \ln\left(\frac{1}{x}\right) - \ln(2) - 1$ we get that

$$\lim_{k\to\infty} 1 + \frac{1-y_k}{\frac{1}{y_1} + \sum\limits_{i=2}^{k} \frac{y_{i-1}}{y_i} + y_k - k} \leq 1 + \frac{1-y_k}{y_k + 1 + \ln\left(\frac{1}{y_k}\right) - \ln(2)}$$

Since $\frac{1-x}{x+1+\ln(\frac{1}{x})-\ln(2)} \geq 0$ if $x \in (0, \frac{1}{2}]$, we get that the maximum of the limit comes from Case 2, which proves the statement of the lemma.

Here we used the fact, that if we compute the maximum of $f(x)$ in the given interval, then we can construct a series of values $y_1, y_2, \ldots, y_k$ which gives an optimal solution for (15). The construction sets y_1 to $\frac{1}{2}$ and defines the geometrical series $y_i = \frac{1}{2}(2y_k)^{\frac{i-1}{k-1}}$, $i = 2, \ldots, k$ which ends with y_k. ■

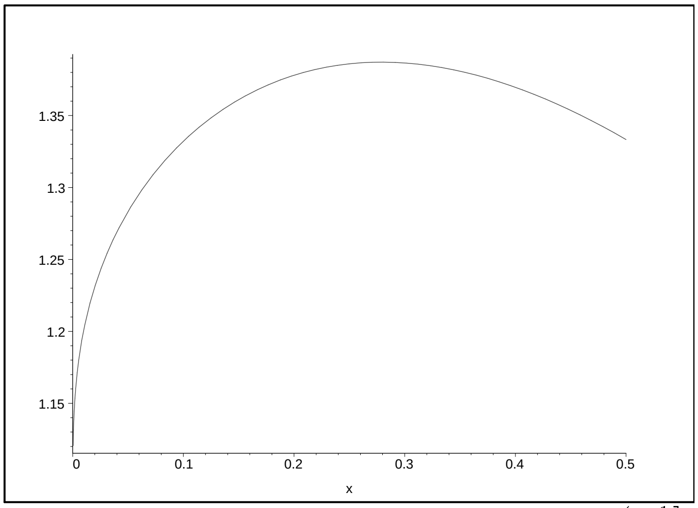

Figure 1. Graph of the function $f(x)$ in the interval $\left(0, \frac{1}{2}\right]$.

Lemma 8 *The maximum of the function* $f(x)$ *is* $1 - \frac{1}{W_{-1}\left(\frac{-2}{e^3}\right)+1} \approx 1.3871$ *in the interval* $\left(0, \frac{1}{2}\right]$, *where* $W_{-1}(x)$ *is the real branch of the Lambert* W *function for which* $W(x) \leq -1$ *holds.*

Proof. The proof is straightforward by taking the derivative of $f(x)$ and investigating the function analytically. ■

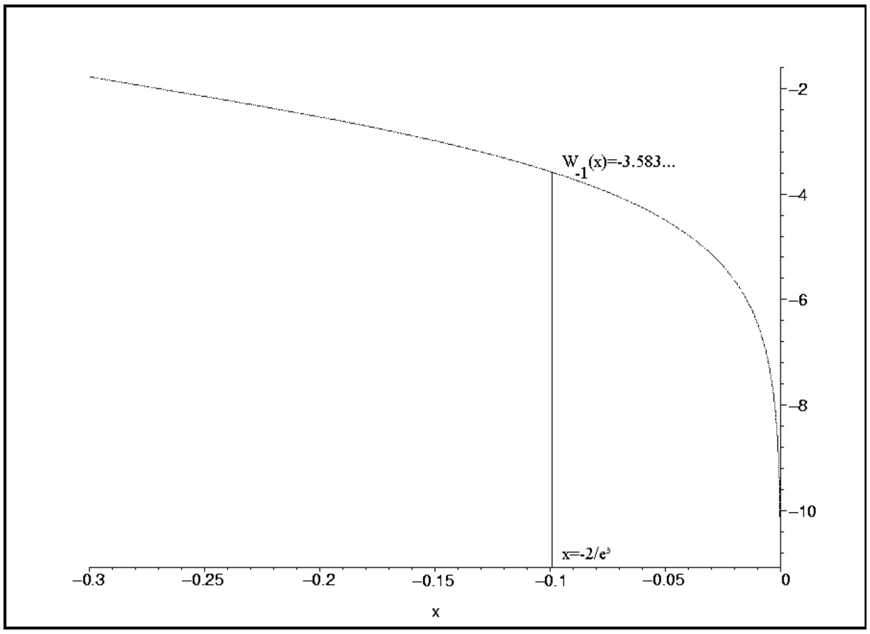

Figure 2. Graph of the function $W_{-1}(x)$ in the interval $[-0.3, 0)$.

5 Conclusions

In this paper we discussed improved lower bounds for the c-repacking version of the classical on-line bin packing problem. Note, however, that these bounds are valid for c-repacking fully-dynamic bin packing algorithms as well. Namely, when deletions are allowed, our construction can be applied in the following way. We first give the list L_0, then insert L_1, and after deleting this, we insert L_2, etc. By examining the list L_0 and the L_0L_1, $L_0L_2, \ldots, L_0L_k$ configurations we obtain the same lower bound with similar arguments as above.

References

[1] Balogh, J., Békési, J., Galambos G. and Markót, M., Analysis of a Non-Linear Optimization Problem Related to Semi-on-line Bin Packing Problems. Manuscript, 2005. Available at http://www.jgytf.u-szeged.hu/˜balogh/babegama20.ps.

[2] Coffman, E.G., Galambos, G., Martello, S., and Vigo, D., Bin Packing Approximation Algorithms: Combinatorial Analysis, in: Handbook of Combinatorial Optimization, eds. D.-Z. Du and P.M. Pardalos, Kluwer, pp. 151-208, 1999.

[3] Coffmann, E.G., Garey, M.R., and Johnson, D.S., Dynamic Bin Packing, SIAM Journal on Computing, 12(2): 227-260, 1983.

[4] Csirik, J. and Woeginger, G.J., On-line Packing and Covering Problems, in: On-line Algorithms, Lecture Notes in Computer Science, eds. A. Fiat and G. Woeginger, Vol. 1442, Berlin, pp. 147-177, 1998.

[5] Fernandez de la Vega, W. and Lueker, G.S., Bin Packing can be Solved Within $1+\varepsilon$ in Linear Time, Combinatorica, 1:349-355, 1981.

[6] Galambos, G. and Woeginger, G.J., Repacking Helps in Bounded Space On-line Bin Packing, Computing, 49: 329-338, 1993.

[7] Gambosi, G., Postiglione, A. and Talamo, M., Algorithms for the Relaxed On-line Bin Packing Model, SIAM Journal on Computing, 30(5): 1532-1551, 2000.

[8] Garey, M.R. and Johnson, D.S., Computers and Intractability (A Guide to the Theory of NP-Completeness). W.H. Freeman and Company, San Francisco, 1979.

[9] Grove, E.F., On-line Bin Packing with Lookahead, in: Proc. of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 430-436, SIAM, 1995.

[10] Gutin, G., Jensen, T., and Yeo, A., Batched Bin Packing, Discrete Optimization, 2(1): 71-82, 2005.

[11] Ivkovič, Z. and Lloyd, E.L., A Fundamental Restriction on Fully Dynamic Maintenance of Bin Packing, Information Processing Letters, 59(4): 229-232, 1996.

[12] Ivkovič, Z. and Lloyd, E.L., Fully Dynamic Algorithms for Bin Packing, Being (Mostly) Myopic Helps, SIAM Journal on Computing, 28(2): 574-611, 1998.

[13] Karmarkar, N. and Karp, R., An efficient approximation scheme for the one-dimensional bin packing problem, in: Proc. of the 23rd Annual Symposium Foundation Computer Science, 312-320, 1982.

[14] Markót, M., private communication, 2005.

[15] Seiden, S.S., On the On-line Bin Packing Problem, Journal of the ACM, 49(5): 640-671, 2002.

[16] Van Vliet, A., An Improved Lower Bound for On-line Bin Packing Algorithms, Information Processing Letters, 43(5): 277-284, 1992.