An Improved Lower Bound for a Semi-on-line Bin Packing Problem*

János Balogh and József Békési

Department of Applied Informatics, Gyula Juhász Faculty of Education,
University of Szeged
E-mail:{balogh,bekesi}@jgypk.u-szeged.hu

Abstract. On-line algorithms have been extensively studied for the one-dimensional bin packing problem. Semi-online property relax the online prescription in such a way that it allows some extra operations or the algorithm knows more (e.g. the optimum value) in advance. In this paper we present an improved lower bound for the asymptotic competitive ratio of any on-line bin packing algorithm which knows the optimum value in advance.

1 Introduction

In computer science the classical one-dimensional bin packing problem is among the most frequently studied combinatorial optimization problems. In its traditional form a list $L = x_1, x_2, \ldots, x_n$ of elements with sizes a_i, $i = 1, \ldots, n$, satisfying $0 < a_i \leq 1$ and an infinite list of unit capacity bins are given. Each element x_i from 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 is also denoted by x_i. The *bin packing problem* involves packing the items to the bins in such a way that as few bins as possible are used.

One specific problem is the so-called *on-line problem*. In this case the elements should be packed in order of their arrival, without any knowledge of the subsequent elements. (Neither the size nor the number of the elements are known.) The packed elements cannot be removed. The algorithms for this problem are called *on-line algorithms*.

In case of *off-line problems* the algorithms have complete information about the list in advance, which they can take into consideration during their operation.

The so-called *semi-on-line algorithms* [2] lie between the well-known on-line and off-line ones. They relax the online prescription in such a way that they allow some extra operations or the algorithm knows more (e.g. the optimum value) in advance. For example the following operations may be allowed: repacking of some items, lookahead of the next several elements, or some kind of pre-ordering, etc.

* This research was supported by Gyula Juhász Faculty of Education, University of Szeged (Project CS-001/2012).

It is well known that finding an optimal packing is NP-hard [6]. Consequently, a large number of papers have been published which look for polynomial time algorithms that find feasible solutions with an acceptable approximation quality. One possibility to measure the performance of an algorithm A is to give its asymptotic competitive ratio (ACR) R_A. For a list L let OPT(L) denote the number of bins in an optimal packing and let $A(L)$ denote the number of bins that algorithm A uses for packing L. If $R_A(l)$ denotes the supremum of the ratios $A(L)/\mathrm{OPT}(L)$ for all lists L with $\mathrm{OPT}(L) = l$, then the asymptotic competitive ratio is defined as

$$R_A := \limsup_{l\to\infty} R_A(l).$$

An equivalent definition is that R_A is the smallest constant, such that there exist a finite constant C, for which $A(L) \leq R_A \cdot OPT(L) + C$ for every list L. It is clear, that $R_A \geq 1$ and the smaller the value R_A is, the better the algorithm is in the worst-case. Of course there are other techniques for measuring the efficiency of an algorithm, for example the average case analysis. Some algorithms may work better in average case(e.g. [3]), others may have better worst-case behaviour.

The current best on-line algorithm with the best known ACR was defined by Seiden [8] in 2002, and it is called *Harmonic++*. Seiden proved that the ACR of his algorithm is at most 1.58889. Harmonic++ belongs to the class of *Super Harmonic* algorithms defined in the same paper. For each algorithm from this class, a lower bound of 1.58333 is valid [7], so $1.58333 \leq R(\text{Harmonic++}) \leq 1.58889$. For on-line algorithms the best known lower bound is $\frac{248}{161} = 1.54037...$ [1].

In this paper we improve the lower bound for the asymptotic competitive ratio of any on-line (one-dimensional) algorithm which knows the optimum value in advance. For the interested reader, we recommend [4], where a motivation of this problem can be found as well. In the bin packing literature, a method for packing patterns is often used to provide lower bounds for different on-line bin packing problems (see e.g. [1, 5, 9]). Here we extend this method. The new construction uses "branching" (input) list sequences.

2 Improved lower bound for a semi-on-line problem

Epstein and Levin considered that semi-on-line problem in [4], when the on-line bin packing algorithm knows the value of the optimum in advance. In their paper they proved a lower bound of 1.30556. Here we raise this lower bound to 1.32312.

The packing pattern technique is described in the literature of bin packing ([1, 5, 9]). It is generally used in the lower bound constructions for bin packing problems. The set of packing patterns of a given sequence of items can be generated. Based on this set, an LP can be constructed and the optimum value of the LP is found for the lower bound of the sequence.

We extend this LP-method in an adaptive way to the so-called "branching" list sequences. The purpose of this extension is to get an automatic method for constructing the LP in a new way, so we provide a technique for computing the

lower bound for these kind of lists. Similar to the case of on-line bin packing algorithms, each list sequence contains a finite number of different lists. The lower bound will be the optimum value of the constructed LP. And like in the on-line case it always exists.

2.1 Packing patterns for branching lists and the construction of the LP

Let us consider r different list sequences, where $r \geq 1$ integer. Denote these by $L_1, \ldots, L_r$, respectively. Let L_i be the concatenation of n_i number of lists, i.e. $L_i = L_{i,1} \ldots L_{i,n_i}$ $(i = 1, \ldots, r)$. Let $L_{i,j}$ also be a concatenation of $n_{i,j} = c_{i,j}n$ elements such a way that each element of $L_{i,j}$ has the same size. Denote this size by $s_{i,j}$ ($c_{i,j} > 0$ and let $0 < s_{i,j} \leq 1$ be real numbers and $j = 1, \ldots, n_i$, $i = 1, \ldots, r$ are integers). Denote the optimum values of the list sequences by $\mathrm{OPT}(L_1), \ldots, \mathrm{OPT}(L_r)$, respectively. In our construction all the optimum values will be equal, whose value will be denoted by N^*.

Let us suppose that a one-dimensional deterministic on-line bin packing algorithm A is executed on the lists $L_1, \ldots, L_r$. The value of N^* is known by A in advance. As a first step we will find the bin types which can be used by the algorithm. Next, we are interested in finding the number of bins of each bin type.

The packing patterns describe how the items of a list sequence L_i $(i = 1, \ldots, r)$ are distributed among the bins. The set of packing patterns for a fixed list L_i is denoted by P^i $(i = 1, \ldots, r)$. The elements of P^i are n_i-dimensional vectors. Each $p = (p_1, \ldots, p_{n_i}) \in P^i$ is a non-negative integer vector, for which $p_1 s_{i,1} + \ldots + p_{n_i} s_{i,n_i} \leq 1$ holds. Here we also define a class of a packing pattern $p \in P^i$: $class(p)$ as the smallest j for which $p_j \neq 0$ in p.

Our procedure is the following:

1. First for each list sequence L_i, we will construct equations for the number of elements in the packing. Each such equation describes all the elements of $L_{i,j}$ can be found in the packing. For each sublist $L_{i,j}$ we have one equation of the form
$$\sum_{p \in P^i} p_j n(p) = n_{i,j}, j = 1, \ldots, n_i, i = 1, \ldots, r,$$
where $n(p)$ is the number of bins packed by pattern p, while packing the elements of the sublist L_i.
Thus for each list L_i we have n_i equations, which means that the total number of such constraints is $n_1 + \ldots + n_r$.
2. Algorithm A packs each list L_i, $i = 1, \ldots, r$, so, for $R(A)$
$$R(A) \geq \limsup_{n \to \infty} \frac{A(L_i)}{OPT(L_i)}$$
holds. It produces one constraint for each list L_i, giving additional r inequalities. We note that there is a difference here compared to the classical on-line algorithms, because we do not add these inequalities for the real sublists of a list L_i (i.e. for the lists $L_{i,1}, \ldots L_{i,t}$ of a list L_i, if $t < n_i$).

It means that

$$\sum_{p \in P^i} n(p) \leq R(A) \cdot \mathrm{OPT}(L_i),\ i = 1, \ldots r.$$

Substituting the previous expression into $\mathrm{OPT}(L_i)$, we get that

$$\sum_{p \in P^i} n(p) \leq R(A) \cdot N^*,\ i = 1, \ldots, r,$$

where the values of $n(p)$-s are non-negative integers.

3. To obtain good lower bounds we have to find some connections among the lists. From this reason we consider a kind of sequences that can have common prefix lists. We call these kinds of lists branching lists. A branching list can be represented by a tree, which can be constructed based on the common prefix portions of the different lists.
Let us now ask the following question of how these connections can appear in the constraints of the LP, which is assigned to the list construction.
Consider all pairs L_i, L_j of the lists ($1 \leq i < j \leq 1$). For every such pair we consider the largest index k, for which $L_{i,1} = L_{j,1}, \ldots, L_{i,k} = L_{j,k}$ holds. It means that if $L_i = L_{i,1} \ldots L_{i,k} L_{i,k+1} \ldots L_{i,n_i}$, then L_j can be written in the form $L_j = L_{i,1} \ldots L_{i,k}\ L_{j,k+1} \ldots L_{j,n_j}$, where $L_{i,k+1} \neq L_{j,k+1}$ (at most one of the lists $L_{i,k+1}$ and $L_{j,k+1}$ can even be empty).
The LP contains additional constraints (equations) for every such i, j pair of indices. More precisely, the number of these additional equations for a given pair i, j is equal to the number of different packing patterns containing an element of the common prefix sublist $L_{i,1} \ldots\ L_{i,k}$. The set $P^{i,k}$ of these packing patterns contains vectors of dimension n_i. This is the subset of P^i that contains the kind of elements of P^i containing an element of the first k sublist of L_i. Therefore

$$P^{i,k} = \{p \in P^i \mid class(p) \leq k\},$$

where, of course, $P^i = P^{i,n_i}$,

$$\sum_{p \in P^{i,k}, p=(p_1,\ldots,p_k,p_{k+1},\ldots,p_{n_i})} n(p)$$
$$= \sum_{q \in P^{j,k}, q=(q_1,\ldots,q_k,q_{k+1},\ldots,q_{n_j})} n(q),$$

for all packing patterns $p \in P_{i,k}$, $q \in P_{j,k}$, $0 \leq i < j \leq 1$, where $n(p)$ and $n(q)$ are the number of bins of type p and q used by the algorithm in the packing of the lists L_i and L_j, respectively. Each equation expresses the equality of the number of bins packed using packing patterns $p \in P^{i,k}$ and $q \in P^{j,k}$, where the first k values are the same, i.e.

$$p = (p_1, \ldots, p_k, p_{k+1}, \ldots, p_{n_i}),$$
$$q = (q_1, \ldots, q_k, q_{k+1}, \ldots, q_{n_j}),$$

where

$$p_1 = q_1, \ldots, p_k = q_k.$$

We note that the number of this kind of constrains may be large.

4. Next, we will consider an algorithm A which gives a minimum value of $R(A)$ subject to all of the above constraints. Our aim is to compute the value

$$R = \min_A \max_{i=1,\ldots,k} \limsup_{n\to\infty} \frac{A(L_i)}{\mathrm{OPT}(L_i)}.$$

This value will be a lower bound for the ACR of any on-line algorithm.

Summarizing the previous constraints, the LP can be given in the following form:

$$\min \ R, \tag{1}$$

subject to

$$\sum_{p\in P^i} p_j n(p) = n_{i,j}, j = 1, \ldots, n_i, i = 1, \ldots, r, \tag{2}$$

$$R \geq \frac{A(L_i)}{\mathrm{OPT}(L_i)} = \frac{\sum\limits_{p\in P^i} n(p)}{N^*}, \ i = 1, \ldots, r, \tag{3}$$

$$\sum_{p\in P^{i,k}, p=(p_1,\ldots,p_k,p_{k+1},\ldots,p_{n_i})} n(p) = \sum_{q\in P^{j,k}, q=(p_1,\ldots,p_k,q_{k+1},\ldots,q_{n_j})} n(q),$$

$$\forall p \in P_{i,k}, q \in P_{j,k}, 0 \leq i < j \leq 1. \tag{4}$$

2.2 The list construction

After these preliminaries we are ready to present an improved lower bound for the bin packing problem with known optimum. Consider the following branching list construction:
$L_1 = L_{1,1}, L_{1,2}$, where
$L_{1,1}$ contains n items, each with size $\frac{1}{40} - 4\varepsilon$, $L_{1,2}$ contains $\frac{39}{40}n$ items with size 1,
$L_2 = L_{2,1}, L_{2,2}, L_{2,3}$, where
$L_{2,1}$ contains n items of size $\frac{1}{40} - 4\varepsilon$, $L_{2,2}$ contains n items of size $\frac{1}{40} + \varepsilon$, $L_{2,3}$ contains $\frac{19}{20}n$ items of size 1,
$L_3 = L_{3,1}, L_{3,2}, L_{3,3}, L_{3,4}$, where
$L_{3,1}$: n items of size $\frac{1}{40} - 4\varepsilon$, $L_{3,2}$: n items of size $\frac{1}{40} + \varepsilon$, $L_{3,2}$: n items of size $\frac{1}{5} + \varepsilon$, $L_{4,3}$: $\frac{3}{4}n$ items of size 1,
$L_4 = L_{4,1}, L_{4,2}, L_{4,3}, L_{4,4}, L_{4,5}$, where
$L_{4,1}$: n items of size $\frac{1}{40} - 4\varepsilon$, $L_{4,2}$: n items of size $\frac{1}{40} + \varepsilon$, $L_{4,3}$: n items of size $\frac{1}{5} + \varepsilon$, $L_{4,4}$: n items of size $\frac{1}{4} + \varepsilon$, $L_{4,5}$: $\frac{1}{2}n$ items of size 1.

$L_5 = L_{5,1}, L_{5,2}, L_{5,3}, L_{5,4}, L_{5,5}$, where
$L_{5,1}$: n items of size $\frac{1}{40} - 4\varepsilon$, $L_{5,2}$: n items of size $\frac{1}{40} + \varepsilon$, $L_{5,3}$: n items of size $\frac{1}{5} + \varepsilon$, $L_{5,4}$: n items of size $\frac{1}{4} + \varepsilon$, $L_{5,5}$: n items of size $\frac{1}{2}$.
Note that ε is a small positive number and n is divisible by 40 in the definitions. It can be seen that after the common portions the lists are filled with padding items 1 to attain the common optimum value. It is not hard to see that for the optimum values $\mathrm{OPT}(L_1) = \ldots = \mathrm{OPT}(L_5) = N^* = n$ holds, so this value can be given for the algorithm in advance. Constructing and solving the LP (1)-(4) leads to optimum value of 1.3231. This means that for the ACR of any on-line algorithms with known optimum value we have a lower bound of 1.3231. This raises the earlier bound of 1.30556 of Epstein and Levin for this problem.

3 Summary and conclusion

Here we presented a new lower bound of 1.3231 for those on-line bin packing problem where the algorithm knows the value of the optimum in advance. We introduced the concept of branching input lists and we gave a new method for constructing the LP. It can be generally used to give lower bounds for bin packing algorithms based on the method of packing patterns. We extended the LP method to the so-called branching lists and of course we hope that this type of construction can be applied to other problems as well. It is a natural question, whether another list can be found, which improves our result.

References

1. J. Balogh, J. Békési, and G. Galambos. New lower bounds for certain classes of bin packing algorithms. *Theoretical Computer Science*, to appear, 2012.
2. E.G. Coffman, G. Galambos, S. Martello, and D. Vigo. Bin packing approximation algorithms: Combinatorial analysis. In *Handbook of Combinatorial Optimization Supplement Volume*, Kluwer Academic Publishers, 151–208, 1999.
3. J. Csirik, D. S. Johnson, C. Kenyon, J. B. Orlin, P. W. Shor, and R. R. Weber. On the sum-of-squares algorithm for bin packing. *J. ACM*, 53(1):1–65, 2006.
4. L. Epstein and A. Levin. On bin packing with conflicts. *SIAM Journal on Optimization*, 19:1270–1298, 2008.
5. G. Galambos and A. van Vliet. Lower bounds for 1-, 2- and 3-dimensional on-line bin packing algorithms. *Computing*, 52:281–297, 1994.
6. M.R. Garey and D.S. Johnson. *Computers and Intractability (A Guide to the theory of NP-Completeness)*, W.H. Freeman and Company, San Francisco, 1979.
7. P. Ramanan, D.J. Brown, C.C. Lee, and D.T. Lee. On-line bin packing in linear time. *Journal of Algorithms*, 10:305–326, 1989.
8. S.S. Seiden. On the on-line bin packing problem. *Journal of the ACM*, 49:640–671, 2002.
9. A. van Vliet. An improved lower bound for on-line bin packing algorithms. *Information Processing Letters*, 43:277–284, 1992.