Online bin stretching with bunch techniques

Michaël Gabay, Vladimir Kotov, Nadia Brauner

Online Bin Stretching with Bunch Techniques

Michaël Gabay[a,*], Vladimir Kotov[b], Nadia Brauner[a]

[a] Grenoble-INP / UJF-Grenoble 1 / CNRS, G-SCOP UMR5272 Grenoble, F-38031, France
[b] Belarusian State University, FPMI DMA department, 4 Nezavisimosti avenue 220030 Minsk Belarus

Abstract

We are given a sequence of items that can be packed into m unit size bins and the goal is to assign these items online to m bins while minimizing the stretching factor. Bins have infinite capacities and the stretching factor is the size of the largest bin. We present an algorithm with stretching factor $26/17 \approx 1.5294$ improving the best known algorithm by Kellerer and Kotov [1] with a stretching factor $11/7 \approx 1.5714$. Our algorithm has 2 stages and uses bunch techniques: we aggregate bins into batches sharing a common purpose.

Keywords: Bin stretching, Multiprocessor scheduling, Online algorithms, Bin packing

1. Introduction

In bin packing problems, a set of items is to be packed into identical bins of size one; the goal is to minimize the number of bins. We are interested in the online variant of this problem: the items arrive consecutively and each of them must be packed irrevocably into a bin, without any knowledge on future items. Recent research has focused on studying scenarios where some information is known in advance.

We consider the online problem where we know in advance that the items can be packed into m bins of size 1. The objective is to pack the items on arrival into m stretched bins, *i.e.* bins of size at most $\beta = 1 + \alpha$ where β is called the stretching factor. Formally speaking, a bin-stretching algorithm is defined to have a stretching factor β if, for every sequence of items that can be assigned to m bins of unit size, the algorithm successfully packs the items into m bins of size at most β. The goal is to find an algorithm with the smallest possible stretching factor.

This problem was introduced by Azar and Regev [2]. They described a practical application of transferring files on a remote system and remarked that this problem is equivalent to the online makespan minimization problem on identical parallel machines with known value of the optimal makespan.

Graham [3, 4] gave the first deterministic online algorithm for this online scheduling problem. He showed that the famous List scheduling algorithm is $(2 - 1/m)$-competitive. A long list of improved algorithms has since been published, the best one is due to Fleisher and Wahl [5].

For the semi-online case, the algorithm is provided with some information on the job sequence or has some extra ability to process it such as decreasing order [4, 6, 7], known total processing time [8, 9, 10, 11], or known number of necessary bins [2] as in our case.

Notice that the bin stretching problem is different from the semi-online scheduling problem with known total processing time. A simple proof of this statement is that Albers and Hellwig [11] proved that 1.585 is a lower bound for the semi-online scheduling problem with known total processing time while Kellerer and Kotov [1] developed an algorithm with stretching factor $11/7 \approx 1.5714 < 1.585$ for the online bin stretching problem. Until recently, 4/3 was the best known lower bound for the bin stretching problem. This bound is

*Corresponding author

Email addresses: michael.gabay@g-scop.grenoble-inp.fr (Michaël Gabay), kotovvm@bsu.by (Vladimir Kotov), Nadia.Brauner@g-scop.grenoble-inp.fr (Nadia Brauner)

obtained with 2 bins, on input (1/3, 1/3, 1) or (1/3, 1/3, 2/3, 2/3) and can be generalized to any number of bins [2]. A better lower bound of $19/14 \approx 1.3571$ for 3 and 4 bins is presented in [12].

Generalizations of the bin stretching problem includes bin stretching with different machine speeds. The case with 2 uniform machines was studied in [13, 14].

In this paper we present an algorithm that uses bunch techniques and provides a stretching factor $26/17 \approx 1.5294$. Recently, Böhm et al. [15] improved the upper bound to 1.5 for any number of bins and to 1.375 for 3 bins, building on the techniques presented in this paper.

1.1. Problem definition and notation

We are given a set of m identical unit size bins and a sequence of n items. Item j has a weight $w_j > 0$ and each item has to be assigned online to a bin. We define the weight of a bin B, denoted by $w(B)$, as the sum of the weights of all items assigned to B. In the course of the algorithm, we define some structures made up of one or several bins. For a given structure S, we denote by $w(S)$ the sum of the weights of all items packed into the bins composing S and $|S|$ is the number of bins in S.

The number m of bins is given as part of the initial input and it is certified that all items can fit into m unit-sized bins. However, we have no more information in the initial input (e.g. the total number of items n is unknown until the end of the input).

We divide the items into 4 disjoint classes as in Table 1 and Figure 1. Items with weight in $(0; \frac{9}{34}]$ are called *tiny*, items in $(\frac{9}{34}; \frac{9}{17}]$ are called *small*, items in $(\frac{9}{17}; \frac{13}{17}]$ are called *medium* and items in $(\frac{13}{17}; 1]$ are called *large*.

Table 1: Item classes

Item class	*tiny*	*small*	*medium*	*large*
Item weight	$(0; 9/34]$	$(9/34; 9/17]$	$(9/17; 13/17]$	$(13/17; 1]$

In the sequel, we design an algorithm with stretching factor $\frac{26}{17}$. Hence, each bin has a capacity $\frac{26}{17}$ and we say that an item j *fits* into a bin B (or equivalently that packing item j into bin B is *feasible*) if $w(B) + w_j \leq \frac{26}{17}$.

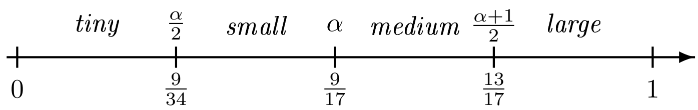

Figure 1: Item types for a stretching factor of $\beta = 1 + \alpha = \frac{26}{17}$

1.2. Algorithm overview

We design a two-stage algorithm. In the first stage, we *open* the bins and create *bunches* which we use to fit the items. In the second stage, we fit the items into the remaining *non-reduced* bins and bunches.

In the algorithm, we use different types of bin structures and qualify them as *open*, *closed* or *reduced*. A structure is a group of one or several bins associated with a qualifier. We say that a bin is *open* if it can be used during current stage of the algorithm. A bin is *closed* once it contains enough items. The *closed* status simply means that the function of the bin changes. Closed bins can be reopened and converted into a new structure anytime. Finally, a bin is *reduced* if it will not be used anymore. Any reduced structure S has the property that the sum of the weights of its items is greater than its number of bins: $w(S) \geq |S|$ and for any bin $B \in S$, $w(B) \leq \frac{26}{17}$. Notice that if all bins have been reduced then there is no item remaining and the stretching factor of the current solution is at most $\frac{26}{17}$.

We denote respectively by sB, mB and lB single bins whose first goal is to contain *small*, *medium* and *large* items. $\mathcal{TB}$ and $\mathcal{LB}$ denote bunches intended to contain respectively *tiny* and *large* items. These bins and bunches can also contain different items as we will see later.

A bunch is a group of 4 bins. The aim of these structures is to help fitting items with more flexibility and then reduce them when structure's total weight is greater than or equal to 4. When a new bunch is created, we first assign a single bin to the bunch, then a second one, a third one and eventually the fourth bin. Once 4 bins have been assigned to a bunch, the bunch is complete and its status changes to *closed*. Otherwise, the bunch is incomplete and is denoted by $\mathcal{TB}^i$ where $i \leq 3$ is the number of bins currently assigned to the bunch.

In the following sections, we describe the different stages of the algorithm and show that any incoming item is packed into a *non-reduced* bin where it fits. This proves Theorem 1.

Theorem 1. *The two-stage algorithm described Sections 2 and 3 has a stretching factor of* $26/17$.

This means that the algorithm never fails and all the weights of the bins are at most $\frac{26}{17}$. In the following sections, we describe the algorithm as a set of priority rules and prove its correctness.

2. Stage 1

At the beginning of the first stage, all bins are empty. Along the first stage, we open bins and organize them into different structures. When an item arrives, Algorithm 1 indicates in which structure it should be packed.

Algorithm 1: Packing item j

```
Let k = 1 and c = class(j)
while j is not packed AND all rules in Table 2 for class c have not been tried do
    if the required structure for rule k of class c exists and is feasible then
        Pack item j according to rule k of class c
        Transform the structure into the new structure given Table 2
    else
        k ← k + 1
if j has not been packed then
    return Fail ;                                    // Goto Stage 2
return Success
```

Algorithm 1 is used while there is no failure. Once the algorithm fails to pack an item, Stage 1 is ended and the algorithm goes into Stage 2.

When the new structure is "*reduced* X *or* Y", it simply means that if $w(B) > 1$ then we reduce B and otherwise, we obtain Y. For instance, if the current item is *small* and an *open sB* exists, then the item is packed into an *open sB*. If the weight of the bin becomes greater than 1, then reduce it. Otherwise the bin remains an *open sB* and further *small* items can still be packed in it. If no *open sB* exists but there is an empty bin, then pack the current item into an empty bin which becomes an *open sB*. If there is no empty bin, then the algorithm goes into Stage 2.

If there is no item remaining, the current solution is feasible and has a stretching factor smaller than or equal to $\frac{26}{17}$. Observe that the empty bin belongs to every set of rules. Hence, Stage 2 cannot be triggered while there is at least one empty bin remaining.

Algorithm 2 explains how items are packed into bunches. *Closed* bunches are made up of 4 bins added one after another. Notice that an *open* $\mathcal{TB}$ bunch contains only *tiny* items and has been assigned at most 3 bins.

Table 2: Stage 1 priority rules

Class	k. Pack in	New structure
large	1. *open* $\mathcal{LB}$	*reduced* $\mathcal{LB}$ or *open* $\mathcal{LB}$
	2. *closed* $\mathcal{TB}$	*open* $\mathcal{LB}$
	3. *open* $\mathcal{TB}^1$	*reduced bin* or *open* lB
	4. *open* $\mathcal{TB}^i$	*reduced bin* and *open* $\mathcal{TB}^{i-1}$
	5. *empty*	*open* lB
medium	1. *open* mB	*reduced bin*
	2. *empty*	*open* mB
small	1. *open* sB	*reduced bin* or *open* sB
	2. *empty*	*open* sB
tiny	1. *open* lB	*reduced bin* or *open* lB
	2. *open* $\mathcal{TB}^3$	*open* $\mathcal{TB}^3$ or *closed* $\mathcal{TB}$
	3. *open* $\mathcal{TB}^i$	*open* $\mathcal{TB}^i$ or *open* $\mathcal{TB}^{i+1}$
	4. *empty*	*open* $\mathcal{TB}^1$

Algorithm 2: Packing *tiny* item j into bunch $\mathcal{TB}^i$

```
// We have an open bunch TB^i composed of bins B_1,...,B_i with i ∈ {1,2,3}
// B_h is the h^th bin assigned to the bunch.
Let h = 1
while j is not packed AND h ≤ i do
    if w(B_h) + w_j ≤ 9/17 then
        Pack item j into B_h
    else
        h ← h + 1
if j has not been packed then
    // h = i + 1
    if there is no empty bin remaining then
        return Fail ;                                        // Goto Stage 2
    Assign an empty bin to the bunch as B_{i+1} and assign j to this bin
else if B_3 contains two items then
    // any two tiny items fit into B_3 with total weight smaller than 9/17
    if there is no empty bin remaining then
        return Fail ;                                        // Goto Stage 2
    Assign an empty bin to the bunch as B_4 and close the bunch
return Success
```

We apply these building rules and obtain the corresponding structures. We give the details of some rules in which there are two structures in the "New structure" field:

- Rule 4 for a *large* item: we pack the item into B_1, the first bin of the bunch. We reduce B_1 and the other bins are renamed: B_2 becomes B_1 and B_3 (if exists) becomes B_2. Rule 3 was not applied, hence $i \geq 2$. Since there is more than one bin in the bunch, $w(B_1) > 9/34$. So any large item fits into B_1 and the weight of B_1 is then greater than 1.

- Rule 2 for a tiny item: we apply the bunch building rules described in Algorithm 2. If the item is packed into B_1 or B_2, we obtain $\mathcal{TB}^3$. Otherwise, we obtain a *closed* $\mathcal{TB}$.

- Rule 3 for a tiny item: we apply the bunch building rules described in Algorithm 2. If the item is packed into B_l with $l \leq i$, we obtain $\mathcal{TB}^i$. Otherwise, we obtain $\mathcal{TB}^{i+1}$. Notice that, since rule 2 for a tiny item was not applied, we have: $i+1 \leq 3$.

Observe that for any $\mathcal{TB}$ bunch, each bin (except B_4) contains at least two items. Denote j and k, the two items in B_3, we have:

$$w(\mathcal{TB}) = (w(B_1) + w_j) + (w(B_2) + w_k) > \frac{18}{17}$$

Once a bunch is *closed*, sort its bins by decreasing order of the weights: $w(B_1) \geq w(B_2) \geq w(B_3) \geq w(B_4) = 0$. Then, the following property holds:

Property 1. *When a bunch is closed, we have:*

$$w(B_1) \geq w(B_2) \geq \frac{6}{17}$$

Proof. $w(B_1)+w(B_2)+w(B_3) > \frac{18}{17}$. Hence the largest weight of a bin is greater than the mean: $w(B_1) \geq \frac{6}{17}$. Both of the two remaining bins are containing at least two items. One precedes the other in the original ordering. W.l.o.g suppose that B_2 was before B_3. Let j and k be two items from B_3. If $w_j \geq \frac{3}{17}$ and $w_k \geq \frac{3}{17}$ then $w(B_3) \geq \frac{6}{17}$. Otherwise, $\min(w_j, w_k) < \frac{3}{17}$ and it did not fit into B_2, hence $w(B_2) > \frac{9}{17} - \frac{3}{17} = \frac{6}{17}$. □

If a *closed* bunch is reopened (as an $\mathcal{LB}$) during Stage 1, items are packed into the first bin in which they fit, by increasing order of bin indices. Note that in a *closed* $\mathcal{TB}$, the remaining capacity in each bin is larger than 1. Hence, we can fit one *large* item into each bin and then $w(\mathcal{LB}) > \frac{18}{17} + 4 \times \frac{13}{17} > 4$ and the bunch can be reduced.

Now it remains to state the reduction rules. For any structure composed of a single bin, reduce it once its weight exceeds 1. $\mathcal{LB}$ structures are reduced once they contain 4 *large* items.

Using the priority rules, one can now easily verify the following properties:

Lemma 1. *Anytime during Stage 1, the following properties hold:*

(i) *all the weights of the bins are smaller than or equal to* $\frac{26}{17}$,
(ii) *there is at most one open* mB,
(iii) *there is at most one open* sB,
(iv) *there is at most one open* $\mathcal{LB}$,
(v) *there is at most one open* $\mathcal{TB}$,
(vi) *there is either no open* lB *or no bunch (neither open nor closed),*
(vii) *(Except rules 2 and 3 for a tiny item) packing an item into the first existing structure is always feasible and results in one of the corresponding structures stated Table 2.*

Note that the exception on property (vii) from Lemma 1 is related to the fact that rules 2 and 3 for a *tiny* item may require an additional empty bin. In such case, if there is no empty bin, the algorithm goes into Stage 2.

Observe that Property (i) from Lemma 1 proves Theorem 1 if the input ends before the algorithm goes into Stage 2.

3. Stage 2

In the second stage, there is no empty bin remaining (except B_4 bins in bunches). We use the remaining space in the open and closed bins and bunches to pack the items. Moreover, there is either no *open* lB or no bunch. We deal with both of these cases separately. In the following, we rely on the following property:

Property 2. *At any step, let* $\mathcal{S}_r$ *be the set of reduced bins,* $|\mathcal{S}_r| = r$. *The total weight of the items which are not packed into* $\mathcal{S}_r$ *is at most* $m - r$.

Proof. If a structure $\mathcal{S}$ is reduced then $w(\mathcal{S}) \geq |\mathcal{S}|$. We sum this up on all reduced structures and obtain: $w(\mathcal{S}_r) \geq r$. Let $\mathcal{I}$ be the set of all items and $\mathcal{I}_r$ the set of items packed into the reduced bins. $w(\mathcal{S}_r) = \sum_{i \in \mathcal{I}_r} w_i = w(\mathcal{I}_r)$. Since all items can be packed into m bins with capacity 1, $w(\mathcal{I}) \leq m$. Hence $w(\mathcal{I}) - w(\mathcal{I}_r) \leq m - r$. □

3.1. All bunches have been reduced

If there is no non-reduced bunch remaining, then there are no *open* $\mathcal{TB}^i$ or *closed* $\mathcal{TB}$ or *open* $\mathcal{LB}$ remaining. At the end of Stage 1, we have some of the following structures:

Reduced bins	*Reduced* $\mathcal{LB}$	
Open lB	*Open* mB (0 or 1)	*Open* sB (0 or 1)

Algorithm 3: Packing item j in Stage 2 (no non-reduced bunch remaining)

if *item* j *fits in an open* lB **then**
  Pack item j into the largest bin *open* lB in which it fits
else
  Pack item j into the largest bin in which it fits
Let B be the bin in which j has been packed
if $w(B) \geq 1$ **then**
  Reduce B

Algorithm 3 indicates how an item is packed during Stage 2. Note that any *small* or *tiny* item can be packed into any non-reduced bin. Hence, while some lB are remaining, *open* mB or *open* sB are only used to pack *medium* or *large* items.

Lemma 2. *If there is no open or closed bunch at the beginning of Stage 2, then Algorithm 3 does not fail and the weight of all bins is smaller than or equal to* $\frac{26}{17}$.

Proof. Suppose that a remaining item j cannot be packed into the remaining open bins. For any non reduced bin B_i, the following inequalities hold:

$$w(B_i) > \frac{9}{17} \tag{1}$$

$$w_j + w(B_i) > \frac{26}{17} \tag{2}$$

Inequality (2), together with the fact that the weight of a non reduced bin is smaller than 1, give $w_j > \frac{9}{17}$. Therefore j is *medium* or *large*.

If there is 0 or 1 open bin remaining, then (2) contradicts Property 2. Hence, there are at least two open bins remaining.

Suppose there is no *open* lB remaining. Then, there are exactly two bins remaining: B_1, an *open* mB and B_2, an *open* sB. We sum up inequalities (1) and (2) and get: $w(B_1) + w(B_2) + w_j > 35/17 > 2$ which contradicts Property 2. Therefore, there are some *open* lB's remaining.

During Stage 1, *tiny* items can only be packed within lB bins or $\mathcal{TB}$ bunches. Since there were no bunches remaining at the beginning of Stage 2, all bunches have been reduced to *reduced* $\mathcal{LB}$. Moreover, there are some *open* lB's remaining. Hence, during Stage 2, all *tiny* items were packed into lB bins. Therefore, *tiny* items have been packed only into bins containing *large* items.

In any feasible solution to the bin packing problem, any bin containing a *large* item can only hold a few additional *tiny* items. Let p be the total number of *large* items and l the number of *large* items already packed.

Denote $B^1, \ldots, B^l$, the bins containing *large* items in the current solution. Because of the preceding remark, we know that $B^{l+1}, \ldots, B^m$ contain no *tiny* item. Hence we can pack j and all items from

$B^{l+1}, \ldots, B^m$ into $m-l$ bins. All bins which are not containing *large* items have been reduced (and hence their weight is greater than 1), except maybe an *open* mB and an *open* sB. From the fact that at least $m-l-2$ bins with no *large* item are reduced, together with inequalities (1) and (2), we obtain:

$$w_j + \sum_{i=l+1}^{m} w(B_i) \geq m - l - 2 + \frac{9}{17} + \frac{26}{17} > m - l$$

This contradicts the feasibility of the packing problem. Indeed, we can pack all *medium*, *small* and any $p-l$ *large* items into $m-l$ bins (one bin for each *large* item and the *medium* and *small* items fit in the other bins). Therefore, there is no such item j. □

We have proved in this case that the algorithm never fails and always returns a solution using at most m bins, filled to at most $\frac{26}{17}$.

Note that if we define the classes as in Figure 1: $(0; \frac{\alpha}{2}]$ (*tiny*), $(\frac{\alpha}{2}; \alpha]$ (*small*), $(\alpha; \frac{1+\alpha}{2}]$ (*medium*) and $(\frac{1+\alpha}{2}; 1]$ (*large*), then all the previous results hold for any $\alpha > 0.5$.

3.2. There are some non-reduced bunches

We now show that Lemma 2 still holds if there are some non-reduced bunches remaining at the end of Stage 1. In this case, there is no *open* lB remaining. Stage 2 starts with some of the following structures:

Reduced bins	*Reduced* $\mathcal{LB}$	
Open mB (0 or 1)	*Open* sB (0 or 1)	
Open $\mathcal{TB}^i$ (0 or 1)	*Open* $\mathcal{LB}$ (0 or 1)	*Closed* $\mathcal{TB}$

During Stage 2, closed bunches are reopened and used to pack some of the remaining items. In the meantime, some *buffer* bins are used to pack the other items. These buffers will receive the smaller items while the larger ones will be packed in the bunches.

Current *buffer* is called $\mathcal{X}$. Along with this buffer, we use up to 3 other single bins: $\mathcal{Z}_1$, $\mathcal{Z}_2$ and $\mathcal{Z}_3$. If there is an *Open* $\mathcal{TB}^i$ at the beginning of Stage 2 we assign its bins to $\mathcal{Z}_1$ and possibly $\mathcal{Z}_2$ and $\mathcal{Z}_3$, by decreasing order of their weights. Whenever we have no $\mathcal{X}$ (Stage 2 is beginning or $\mathcal{X}$ is reduced), the first existing structure among the following becomes $\mathcal{X}$:

$$\textit{open } sB, \ \textit{open } mB, \ \mathcal{Z}_3, \ \mathcal{Z}_2, \ \mathcal{Z}_1, \ \textit{closed } \mathcal{TB}$$

In all but the last case, we get $\mathcal{X}$ by renaming a bin. In the last case, we denote by B_1, B_2, B_3, B_4 the bins from the bunch, $w(B_1) \geq w(B_2) \geq w(B_3) \geq w(B_4)$. We assign: $\mathcal{X} \leftarrow B_4$, $\mathcal{Z}_1 \leftarrow B_1$, $\mathcal{Z}_2 \leftarrow B_2$ and $\mathcal{Z}_3 \leftarrow B_3$ and the bunch is disbanded.

If we cannot get a new $\mathcal{X}$, then only a few bins are remaining. Stage 2 is terminated and the algorithm goes into a last stage, detailed in Section 3.2.2.

During Stage 2, an additional type of bunch, denoted by $\mathcal{MB}$ is used. The main purpose of these bunches is to receive *medium* items.

The process is then very similar to Stage 1: items are packed into bins according to priority rules and bins are reduced. Priority rules are given Table 3. There is however a slight difference with Table 2: it should be read as "Pack item j into structure $\mathcal{S}$ if $\mathcal{S}$ exists and packing item j into $\mathcal{S}$ is feasible and results in the new structure indicated in Table 3". This difference only concerns rule (1) for *large* items. Indeed: $\mathcal{Z}_1$ was part of a (possibly open) bunch. Therefore, at the end of Stage 1, its weight was smaller than $9/17$ and any item can be packed into $\mathcal{Z}_1$. However, we only pack an item into $\mathcal{Z}_1$ if we can reduce it afterwards. If $\mathcal{Z}_1$ is reduced, then $\mathcal{Z}_1 \leftarrow \mathcal{Z}_2$ and $\mathcal{Z}_2 \leftarrow \mathcal{Z}_3$ (if exists).

When an item is assigned to a single bin structure, if the weight of the bin becomes greater than 1, then the bin is reduced.

When an item is assigned to an *open* $\mathcal{LB}$, we try to pack it into B_3, then B_2, B_1 and eventually B_4. Once B_4 contains an item, we reduce the bunch. As seen in Stage 1, the weight of the structure is greater than 4. When a *medium* item is assigned to a *closed* $\mathcal{TB}$, it is packed into B_3. When an item is assigned to an

Table 3: Stage 2 priority rules

Item	Pack in	New structure
large	1. $\mathcal{Z}_1$	*reduced bin*
	2. *open* $\mathcal{LB}$	*reduced* $\mathcal{LB}$ or *open* $\mathcal{LB}$
	3. *closed* $\mathcal{TB}$	*open* $\mathcal{LB}$
	4. $\mathcal{X}$	*reduced bin* or $\mathcal{X}$
medium	1. *open* mB	*reduced bin*
	2. $\mathcal{X}$	*reduced bin* or $\mathcal{X}$
	3. *open* $\mathcal{MB}$	*reduced* $\mathcal{MB}$ or *open* $\mathcal{MB}$
	4. *closed* $\mathcal{TB}$	*open* $\mathcal{MB}$
{*small*, *tiny*}	1. $\mathcal{X}$	*reduced bin* or $\mathcal{X}$
	2. *open* $\mathcal{MB}$	*reduced* $\mathcal{MB}$ or *open* $\mathcal{MB}$

open $\mathcal{MB}$ we try to pack it into B_3, then B_2 and eventually B_4. Since B_4 was empty at the end of Stage 1, we can pack any two *medium* items into B_4. When B_4 contains 2 items, we reduce B_2, B_3, B_4 and $\mathcal{X}$ and $\mathcal{X} \leftarrow B_1$. The following property shows that these bins can indeed be reduced:

Property 3. *Once B_4 from an open $\mathcal{MB}$ contains two items, $w(\mathcal{X}) + w(B_2) + w(B_3) + w(B_4) > 4$.*

Proof. During Stage 2, at least one item j which did not fit into B_3 has been packed into B_2. Hence, by Property 1:

$$\begin{aligned} w(B_3) + w(B_2) &= (w(B_3) + w_j) + (w(B_2) - w_j) \\ &> 26/17 + 6/17 = 32/17 \end{aligned}$$

Therefore, $\max(w(B_3), w(B_2)) > 16/17$. Moreover, B_4 contains two items k and l (with l the last item packed). Neither k, nor l fit into B_3 or B_2 and l does not fit into $\mathcal{X}$. Hence:

$$\begin{aligned} & w(\mathcal{X}) + \min(w(B_3), w(B_2)) + w(B_4) \\ & \quad \geq (w(\mathcal{X}) + w_l) + (\min(w(B_3), w(B_2)) + w_k) \\ & \quad > 26/17 + 26/17 = 52/17 \end{aligned}$$

Eventually, summing this up with $\max(w(B_3), w(B_2))$ gives:

$$w(\mathcal{X}) + w(B_2) + w(B_3) + w(B_4) > 4$$

□

Observe that there is no assumption on the classes of the items packed into $\mathcal{X}$, B_2 and B_3 in Property 3.

3.2.1. Termination stage

Stage 2 is completed, either when the input is over or no packing rule is feasible (or we cannot get a new $\mathcal{X}$ – in such case, refer to Subsection 3.2.2). In the following, we consider the different cases and show that we can always fit remaining items into non-reduced bins with a 26/17 stretching factor.

If the algorithm finishes before an item cannot be packed according to priority rules, then all items have been packed and none of the bins capacities exceeds 26/17. If all bins have been reduced, then the sum of all the weights of the bins is greater than m and hence all items have been packed.

Otherwise, no rule can be applied to pack the current item. Table 4 sums up the possibly remaining structures depending on the current item. Note that for a *large* item, if $\mathcal{Z}_2$ exists, then $w(\mathcal{Z}_1) > \frac{9}{34} > \frac{4}{17}$ and since $w(\mathcal{Z}_1) \leq \frac{9}{17}$, we can apply rule 1 for a *large* item. Hence if current item is *large* and no rule can be applied, then there is no $\mathcal{Z}_2$ remaining. The reader can easily verify remaining configurations for the other classes of items.

Table 4 does not take *open mB* into account. We deal with this case as follows: if an *open mB* is remaining, then no *medium* item came during Stage 2. Hence, there is no $\mathcal{MB}$ bunch. Moreover, the current item j is *large* since any *tiny* or *small* item would fit into $\mathcal{X}$ and any *medium* into *open mB*. Therefore, there is no $\mathcal{Z}_2$. The remaining bins are $\mathcal{X}$, *open mB* and possibly $\mathcal{Z}_1$. The remaining items are packed according to Subsection 3.2.4.

Table 4: Remaining structures depending on the current item

Current item	Remaining bins
large	*open* $\mathcal{MB}$, $\mathcal{X}$, $\mathcal{Z}_1$
medium	*open* $\mathcal{LB}$, $\mathcal{X}$, $\mathcal{Z}_1$, $\mathcal{Z}_2$, $\mathcal{Z}_3$
small	*open* $\mathcal{LB}$
tiny	*open* $\mathcal{LB}$

If a bunch is remaining, we denote its bins by B_1, B_2, B_3 and B_4. Depending on the current configuration, we reduce some of the remaining bins as detailed in Algorithm 4.

Algorithm 4: Termination Stage

```
if there is an open LB containing 3 large items and no Z3 then                  // case 1
    Reduce B1, B2 and B3
else if there is an open LB containing 3 large items and Z3 then                // case 2
    if current item j fits into X then
        Pack j into X
        if w(X) ≥ 1 then
            Reduce X, B1, B2 and B3
    else
        Pack j into Z1
        Reduce X, Z1, B1, B2 and B3
else if there is an open MB remaining then                                      // case 3
    if current item j fits into B2 then
        Pack j into B2
        Resume Stage 2
    else
        Pack j into B1
        Reduce B1, B2 and B3
else if there is an open LB containing 1 or 2 large items remaining then        // case 4
    Consider this bunch as an open MB and resume Stage 2
else                                                                            // case 5
    // There is no bunch and at most 4 remaining bins
    Use the rules given in Section 3.2.3 to 3.2.5 to terminate
```

In the following, we explain why Algorithm 4 works and show that the remaining bins are in one of the configurations treated in the next subsections.

1. If there is an *open* $\mathcal{LB}$ containing 3 *large* items and no $\mathcal{Z}_3$. Reducing B_1, B_2 and B_3 is feasible because the bunch contains 3 *large* items and was a *closed* $\mathcal{TB}$ bunch before being reopened; hence its weight

was greater than 18/17. Therefore:

$$w(B_1) + w(B_2) + w(B_3) \geq 18/17 + 3 \times 13/17 = 57/17 > 3$$

Then, we have at most 4 bins remaining: $B_4, \mathcal{X}, \mathcal{Z}_1$ and $\mathcal{Z}_2$.

2. If there is an *open* $\mathcal{LB}$ containing 3 *large* items and $\mathcal{Z}_3$, we pack all coming items into $\mathcal{X}$ until it is reduced and then we reduce B_1, B_2 and B_3 as previously. Otherwise, current item j does not fit into $\mathcal{X}$. Since $\mathcal{Z}_3$ exists, we can use Property 1 for $\mathcal{Z}_1$:

$$w_j + w(\mathcal{X}) + w(\mathcal{Z}_1) + (w(B_1) + w(B_2) + w(B_3)) >$$
$$26/17 + 6/17 + 57/17 > 5$$

We pack j into $\mathcal{Z}_1$ and reduce $\mathcal{X}$, $\mathcal{Z}_1$, B_1, B_2 and B_3. Then, $\mathcal{Z}_1 \leftarrow \mathcal{Z}_2$, $\mathcal{Z}_2 \leftarrow \mathcal{Z}_3$ and we have exactly 3 bins remaining: $\mathcal{Z}_1$, $\mathcal{Z}_2$ and B_4.

3. If there is an *open* $\mathcal{MB}$ remaining, then j (the current item) is *large*. If j fits into B_2 we pack it into B_2 and resume with priority rules. Property 3 still holds. Otherwise, B_2 contains an item which does not fit into B_3. Hence, $w(B_2) + w(B_3) > \frac{26}{17} + \frac{6}{17} = \frac{32}{17}$. Once j is packed into B_1, $w(B_1) + w(B_2) + w(B_3) > \frac{6}{17} + \frac{13}{17} + \frac{32}{17} = 3$ and we can reduce B_1, B_2 and B_3. There are at most 3 remaining bins: B_4, $\mathcal{X}$ and $\mathcal{Z}_1$.
4. If there is an *open* $\mathcal{LB}$ containing 1 or 2 *large* items remaining, we consider this bunch as an *open* $\mathcal{MB}$ and keep on applying priority rules and eventually previous point (3).
5. Otherwise, there is no bunch. There are at most 4 bins remaining: $\mathcal{X}, \mathcal{Z}_1, \mathcal{Z}_2$ and $\mathcal{Z}_3$.

After these reductions, we have at most 4 bins remaining. Let b be the number of remaining bins. In each case, we explain how to use the remaining bins and then consider j, an item which does not fit into any of the remaining bins. We show that w_j plus the sum of the weights of the remaining bins is strictly greater than b, contradicting Property 2.

The cases with 0 or 1 bin remaining are trivial so we only deal with the other cases.

3.2.2. We cannot get a new $\mathcal{X}$

If we cannot get a new $\mathcal{X}$, then remaining bins are possibly an *open* $\mathcal{MB}$ and an *open* $\mathcal{LB}$. We keep on applying priority rules. However, when an item is packed into *open* $\mathcal{MB}$, we try to pack it into B_3, then B_2, then B_1 and eventually B_4. Hence, if the *open* $\mathcal{MB}$ is reduced, its 4 bins are reduced.

Once there is a single structure remaining, if it is the *open* $\mathcal{LB}$, then we reduce the bins and finish as presented Subsection 3.2.1.

Otherwise there is an *open* $\mathcal{MB}$ remaining. We keep on applying priority rules and suppose some item j cannot be packed.

The item j cannot be packed. Hence B_1 and B_4 both contain an item which fits into neither B_2, nor B_3. Denote those items k and l. By Property 1: $w(B_1) - w_k \geq \frac{6}{17}$. Moreover, Property 3 holds. Therefore, B_4 contains a single item. Therefore, either l or j (or both) is *large*. Without loss of generality, suppose j is *large*, then:

$$\begin{aligned} w(B_1) + w(B_2) + w(B_3) + w(B_4) + w_j \\ &\geq (w(B_1) - w_k) + (w(B_2) + w_k) + (w(B_3) + w_l) + w_j \\ &> 6/17 + 26/17 + 26/17 + 13/17 \\ &> 4 \end{aligned}$$

Which is a contradiction.

Table 5: Renaming scheme

New names	L_1	L_2	L_3	L_4
Old names	$\mathcal{X}$	B_4	$\mathcal{Z}_2$	$\mathcal{Z}_1$
	$\mathcal{X}$	$\mathcal{Z}_3$	$\mathcal{Z}_2$	$\mathcal{Z}_1$

3.2.3. 4 bins remaining

If there are 4 remaining bins, the possibly remaining bins are detailed Table 5. We rename those bins L_1, L_2, L_3 and L_4. Note that $w(L_2), w(L_3), w(L_4) \leq \frac{9}{17}$ at the beginning of this step. Hence we can fit at least one item in any of those three bins.

Pack any fitting item into L_1, otherwise L_2, then L_3 and eventually into L_4. Suppose j is an item which does not fit into any of the remaining bins. Denote k_i the last item packed into L_i and observe that, for $i = 2, 3, 4$, k_i does not fit into L_f for all $f < i$.

If the weight of a bin is greater than 1, then:

$$\begin{aligned} &w(L_1) + w(L_2) + w(L_3) + w(L_4) + w_j \\ &\quad > 1 + 26/17 + 26/17 \\ &\quad > 4 \end{aligned}$$

Otherwise, all the weights of the bins are smaller than one. Hence $w_j > \frac{9}{17}$. Moreover, at the beginning of this step, $w(L_3) + w(L_4) > \frac{9}{17}$.

$$\begin{aligned} &w(L_1) + w(L_2) + w(L_3) + w(L_4) + w_j \\ &\quad \geq (w(L_1) + w_{k_3}) + (w(L_2) + w_{k_4}) + \\ &\quad\quad (w(L_3) + w(L_4) - w_{k_3} - w_{k_4}) + w_j \\ &\quad > 26/17 + 26/17 + 9/17 + 9/17 \\ &\quad > 4 \end{aligned}$$

Which is a contradiction.

3.2.4. 3 bins remaining

If there are 3 bins remaining, then $\mathcal{Z}_1$ is among them and $w(\mathcal{Z}_1) \leq \frac{9}{17}$. Rename it L_3 and the other bins are renamed L_1 and L_2. Pack any fitting item into L_1, otherwise L_2 and eventually L_3. Suppose that the item j does not fit into any of them and let k be the last item packed into L_3. There is at least one such item since $w(\mathcal{Z}_1) \leq \frac{9}{17}$ in the beginning.

$$\begin{aligned} &w(L_1) + w(L_2) + w(L_3) + w_j \\ &\quad \geq (w(L_1) + w_j) + (w(L_2) + w_k) \\ &\quad > 26/17 + 26/17 \\ &\quad > 3 \end{aligned}$$

Which is a contradiction.

3.2.5. 2 bins remaining

In this case, denote one bin by L_1 and the other bin by L_2. Pack any fitting item into L_1, otherwise into L_2. If j does not fit into L_2, then $w(L_2) > \frac{9}{17}$.

$$w_j + w(L_1) + w(L_2) > 26/17 + 9/17 > 2$$

Which is a contradiction.

4. Complexity

We represent a bin and its content using a stack plus its current weight and use a dedicated data structure (a stack) for each kind of structure used in the algorithm. The overall space used is $\mathcal{O}(m)$.

In order to pack any given item during Stage 1, we need to check its class and try to pack it in at most 5 different structures with at most 3 bins tested for each one. Hence, any item is packed in $\mathcal{O}(1)$ time. Therefore the overall complexity of the first stage is bounded by $\mathcal{O}(n)$.

During Stage 2, we need to sort the structures. Each structure has at most 4 bins. Hence, a structure is sorted in $\mathcal{O}(1)$ time and we have at most $\frac{m}{4}$ structures to sort. Therefore, we sort all of them in $\mathcal{O}(m)$ time. In order to pack any item, we need to check its class and try at most 4 different structures. Hence, any item is packed in $\mathcal{O}(1)$ time and the overall complexity of this stage is bounded by $\mathcal{O}(n)$.

Same goes for the termination stage. Moreover, additional operations, like renumbering the bins, are performed but there is a fixed number of different additional operations and all of them are performed in constant time.

Furthermore, when $m \geq n$, at most n bins are used. Hence, the overall time and space complexity of the algorithm is $\mathcal{O}(n)$.

5. Summary and future work

The presented algorithm has a stretching factor of $\frac{26}{17}$ and runs in linear time. Notice that this bound is tight with the input $m = 2$ and the items: $\{\frac{13}{17}, \frac{13}{17}\}$.

The techniques of combining bins into bunches with certain properties and analyzing the bunches has been successfully applied to other online and offline packing problems, see e.g. [16, 17].

It seems reasonable to hope that better worst-case behavior can be achieved by refining this approach. Based on this scheme, it might be possible to reduce the gap between lower and upper bound for both known total sum and bin-stretching problems. Improving lower bounds is also a challenging task.

Acknowledgments

This research has been partially supported by project ICS No 5379 and Belorussian BRFFI grant (Project F13K-078). The research of the first and the third author have been partially supported by the LabEx PERSYVAL-Lab (ANR–11-LABX-0025).

References

[1] Hans Kellerer and Vladimir Kotov. An efficient algorithm for bin stretching. *Operations Research Letters*, 41(4):343–346, 2013.

[2] Yossi Azar and Oded Regev. On-line bin-stretching. *Theoretical Computer Science*, 268(1):17–41, 2001.

[3] Ronald L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45(9):1563–1581, 1966.

[4] Ronald L. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, 17(2):416–429, 1969.

[5] Rudolf Fleischer and Michaela Wahl. On-line scheduling revisited. *Journal of Scheduling*, 3(6):343–353, 2000.

[6] Steve Seiden, Jiří Sgall, and Gerhard Woeginger. Semi-online scheduling with decreasing job sizes. *Operations Research Letters*, 27(5):215–221, 2000.

[7] T.C. Edwin Cheng, Hans Kellerer, and Vladimir Kotov. Algorithms better than LPT for semi-online scheduling with decreasing processing times. *Operations Research Letters*, 40(5):349–352, 2012.

[8] Hans Kellerer, Vladimir Kotov, Maria Grazia Speranza, and Zsolt Tuza. Semi on-line algorithms for the partition problem. *Operations Research Letters*, 21(5):235–242, 1997.

[9] Enrico Angelelli, Á.B. Nagy, Maria Grazia Speranza, and Zsolt Tuza. The on-line multiprocessor scheduling problem with known sum of the tasks. *Journal of Scheduling*, 7(6):421–428, 2004.

[10] T.C. Edwin Cheng, Hans Kellerer, and Vladimir Kotov. Semi-on-line multiprocessor scheduling with given total processing time. *Theoretical computer science*, 337(1):134–146, 2005.

[11] Susanne Albers and Matthias Hellwig. Semi-online scheduling revisited. *Theoretical Computer Science*, 443:1–9, 2012.

[12] Michaël Gabay, Nadia Brauner, and Vladimir Kotov. Computing Lower Bounds for Online Optimization Problems: Application to the Bin Stretching Problem. *HAL preprint hal-00921663*, 2013.

[13] György Dósa, Maria Grazia Speranza, and Zsolt Tuza. Two uniform machines with nearly equal speeds: unified approach to known sum and known optimum in semi on-line scheduling. *Journal of Combinatorial Optimization*, 21(4):458–480, 2011.

[14] C.T. Ng, Zhiyi Tan, Yong He, and T.C. Edwin Cheng. Two semi-online scheduling problems on two uniform machines. *Theoretical Computer Science*, 410(8):776–792, 2009.

[15] Martin Böhm, Jiří Sgall, Rob van Stee, and Pavel Veselý. Better algorithms for online bin stretching. *arXiv preprint arXiv:1404.5569*, 2014.

[16] Hans Kellerer and Vladimir Kotov. An approximation algorithm with absolute worst-case performance ratio 2 for two-dimensional vector packing. *Operations Research Letters*, 31(1):35–41, 2003.

[17] Luitpold Babel, Bo Chen, Hans Kellerer, and Vladimir Kotov. Algorithms for on-line bin-packing problems with cardinality constraints. *Discrete Applied Mathematics*, 143(1):238–251, 2004.