Online bin packing with delay and holding costs

Lauri Ahlroth[*], André Schumacher, Pekka Orponen

Aalto University School of Science, Department of Information and Computer Science and

Helsinki Institute for Information Technology HIIT

Abstract

We consider online bin packing where in addition to the *opening cost* of each bin, the arriving items collect *delay costs* until their assigned bins are released (closed), and the open bins themselves collect *holding costs*. Besides being of practical interest, this problem generalises several previously unrelated online optimisation problems. We provide a general online algorithm for this problem with competitive ratio 7, with improvements for the special cases of zero delay or holding costs and size-proportional item delay costs.

Keywords: competitive analysis, online algorithm, bin packing, rent-to-buy

1. Introduction

Online bin packing and its variants can be used to model a large number of application scenarios, such as transportation of goods, telecommunications, and jobshop scheduling. Consider for instance the task of shipping items from a shipping depot (factory, warehouse) to another destination. Posting a shipment incurs a cost, and so the overall cost of the task can be reduced by including several items in the same shipment. However, shipping containers have bounded capacity, so only a limited number of items can be sent together, where the number may depend on the sizes of the items. This example scenario, and many similar ones, can be formulated in terms of the classical NP-complete bin packing problem. Bin packing and its online variant, in which items must be assigned to bins upon their arrival without knowledge of future arrivals, have been studied extensively [1, 2, 3, 4].

Continuing with the example, one may consider the case where the items are perishable goods that lose value while waiting, or collect some other type of delay cost over time until they arrive at their destination. If one further accounts for the capital costs or rents of the containers themselves, then the depot should also not let the containers wait too long before shipping them. This is an example for an often encountered situation that calls for layered decisions. Items arriving in an online fashion need to be packed into bins (containers, packets, machines), where both the items and the bins themselves collect cost until they are released for further processing (shipping, transmission, tooling).

We study versions of the online bin packing problem that can be used to model these types of scenarios. In addition to the *opening cost* per each bin used, we assume that the arriving items collect *delay costs* while waiting for their assigned bins to be released, and the bins themselves collect corresponding *holding costs*. In addition to transportation problems, similar issues are frequently encountered e.g. in the context of communication networks, where packets collect delay while waiting for transmission and transmission slots have limited capacity.

We study online algorithms within the context of *competitive analysis*. An (online) algorithm $\mathcal{A}$ has performance ratio $\phi \geq 1$ for a given problem instance if it achieves a cost of ϕ times the optimal (offline) cost. The supremum of these ratios over all possible instances is called the *competitive ratio* of $\mathcal{A}$. In the model of this paper, we show in Appendix A that under reasonable assumptions for online algorithms the value of the usual *asymptotic* competitive ratio is equal to the *strict* competitive ratio defined above.

Besides its direct interpretation in terms of online bin packing, the problem we study can be considered as an extension of the *TCP acknowledgement problem* [5, 6, 7, 8]. In this setting one seeks to acknowledge a sequence of arriving packets with as few acknowledgement messages as possible, taking into account the delay accumulated between the arrival times and the eventual acknowledgement of each packet. There is no restriction on the number of packets that can be acknowledged by a single message, so that this setting corresponds to a single-bin *uncapacitated* version of our problem. The TCP acknowledgement problem is also closely related to the well-known *ski rental problem* [9]. Both problems have an asymptotically tight upper and lower bound of 2 for their competitive ratio [5].

In the context of operations research, the discrete-time version of the TCP acknowledgement problem is also known as the *single-commodity economic lot-sizing problem*. In this setting, the problem was first studied by Wagner and Whitin [10],

[*] Corresponding address: Aalto University School of Science, Department of Information and Computer Science, P.O.Box 15400, FI-00076 Aalto, Finland. Tel. +358 9 47025141, fax: +358 9 47023277

Email addresses: lauri.ahlroth@aalto.fi (Lauri Ahlroth), andre.schumacher@aalto.fi (André Schumacher), pekka.orponen@aalto.fi (Pekka Orponen)

who also proposed a polynomial time algorithm solving the offline version to optimality. Variations of this problem have been studied with and without capacity constraints [11, 12], typically however only in an offline setting. It is interesting that the frequently used Part-Period-Balancing (PPB) heuristic for the lot-sizing problem [13] is essentially identical to the online algorithm given in [5] for the TCP acknowledgement problem, where the *holding cost* due to inventory kept in a warehouse takes the role of item delay cost and *ordering cost* is replaced by transmission cost.

When comparing results between the two families of related problems, online bin packing and TCP acknowledgement, one notes that for online bin packing the best possible asymptotic competitive ratio is between 1.5403 [14] and 1.5889 [2]. This is in contrast to the TCP acknowledgement problem, which has an asymptotic lower bound of 2, matching its upper bound [5]. It appears that the combination of these two problems, viz. taking into account both the opening costs of capacity-limited bins and costs induced by item delays, has not been previously explored.

We propose a general type of algorithm that is inspired by the PPB heuristic. Following the general principle of the heuristic, the algorithm simply releases a bin when the sum of its accumulated delay and holding cost approximately equals the bin opening cost. The assignment of items to bins is handled via one of the traditional bin packing heuristics, whose operation is limited to opening a new bin and assigning items to bins that have not yet been released. In general, the algorithm may decide to keep multiple bins open at the same time.

For the special case in which there are only delay costs but no holding costs, we obtain slightly improved results under the additional assumption that items collect delay cost at a rate that is proportional to their size. This second algorithm is inspired by the *Next-Fit* bin packing heuristic [4], which we extend to our new problem. Next-Fit maintains a single *active* bin, to which it assigns newly arriving items until bin is full and a new bin must be opened, which then becomes the active bin. For both algorithms, we establish constant worst-case competitive ratios for our generalisation of the bin packing problem. The actual values of the constants obtained are summarised at the end of the paper in Table 1.

The rest of the paper is organised as follows. Section 2 formalises the problem and introduces our model. Section 3 introduces a class of online algorithms and proves constant upper bounds on their competitive ratio, including improved results for special cases. We then turn to the case when the delay cost that an item collects is proportional both to its size and the time it spends waiting for its bin to be released. Section 4 gives performance results for a different algorithm that hold under this assumption and improve on our results for the general case. Finally, Section 6 presents our conclusions and outlines future work. In Appendix A we further show that the asymptotic and strict competitive ratios have mutually equal values for most intuitive algorithm classes.

2. The model

Our model resembles the classical online bin packing problem in the sense that *items* arrive in an online fashion and have to be assigned to *bins* upon their arrival. Each item has a *size* and the *load* of a bin (the total size of items assigned to it) must not exceed its *capacity*. As also in the classical setting, we penalise algorithms that choose to *open* many bins by charging an opening cost. However, our model differs from the classical one in that the items await further processing after they were assigned to their bin at their *arrival time*. More precisely, we require that each opened bin needs to eventually be *released*, thereby making it unavailable for the assignment of further items arriving in the future. This variation is motivated by various problems from the context of communication and transportation networks.

In the setting of our problem, the most natural way to model item arrivals is in the model that is typically referred to as *online-time* model in the scheduling literature (e.g., see [15]). In the online-time model the item arrival times correspond to job release dates, and the problem instance is revealed to the algorithm over time as jobs are released (not to be confused with the release of bins in our model). The online-time model is different from the *online-list* model, which is typically used in the context of bin packing but does not immediately allow for the introduction of costs relating to time since the instance is solely defined by the sequence of items (and their sizes) arriving.

Any algorithm for this problem needs to split the sequence of items into some number of opened bins and decide on a *release time* for each of these, which then determines the total delay and holding cost incurred by the bin. We denote the cost for opening a new bin by B. One could w.l.o.g. assume $B = 1$ by scaling all costs by $1/B$. We however prefer to retain the explicit reference to B in order to indicate the dependence of the various algorithms on this parameter.

We study the effect of two different types of additional costs on the performance of algorithms. Our first variation assumes that each item collects a *delay cost* from time of item arrival to release of the item. Different items may in general have different delay cost functions. Fix an item k and denote by t the time elapsed from the arrival of item k. The accumulated delay cost function $D_{(k)}(t)$ is assumed to satisfy the following properties:

1. $D_{(k)}(0) = 0$,
2. $D_{(k)}(\cdot)$ is non-decreasing and continuous,
3. $\sup_{t>0} D_{(k)}(t) > B$.

The third assumption guarantees that an optimal solution releases items instead of keeping them in an open bin forever. This notion of delay cost allows us to model a great variety of settings, for example these in which items belong to classes that collect delay costs at different rates, depending on a notion of urgency of being released. We point out a *bilinear* special case where the delay cost is

$$D_{(k)}(t) = d \cdot s_k \cdot t,$$

with s_k being the size of item k and d a constant common to all items.

The second cost variation assumes that each bin collects a *holding cost* from the time it was opened until the time it was released. The holding cost is assumed to have a linear structure: there is a constant $h > 0$ such that bin i collects holding cost $H_i = h \cdot t$ when it has been open for t time units. Observe that the holding cost also satisfies all three properties of the delay cost function given above. The third cost variation of our problem, that is when both delay and holding cost are present, simply considers the sum of both delay and holding costs.

Similar cost functions have been studied by Dooly et al. [5] for the TCP acknowledgement problem. A special case of our first variant, where $D_{(k)}(t) = d \cdot t$ for some constant d, is referred to as f_{sum} and the second holding cost variant corresponds to the f_{max} cost function, while their combination satisfies the conditions on the cost function f_{lat} defined in [5]. We use this fact for obtaining some lower bounds by corollaries from the TCP acknowledgement problem.

Following the classical bin packing problem, we assume uniform bin capacities, which means that we consider bins with capacity 1. Let l_i^t denote the load of bin i at time t, which is defined as the sum of the size of all items assigned to i up to and including time t. Due to the capacity constraint we thus require that $l_i^t \leq 1$ for each bin i at any time t. Also let $l_i := \sup_t l_i^t$ denote the final load present in bin i after the whole input is processed.

No more items can be assigned to i that arrive after its release time. More specifically, let T_i be the time interval between the opening time of bin i and its release time (endpoints included). We can always assume that the opening time is equal to the arrival time of the first item that was assigned to bin i. During T_i bin i collects holding cost of $h\,|T_i|$ and all items that are assigned to it collect delay cost.

Let further D_i denote the total delay cost contributed by the items placed in i. If I is a set of bins opened in a solution, $D_I := \sum_{i \in I} D_i$ and $H_I := \sum_{i \in I} H_i$. The notion $D_i(T_{i'})$, for some bin i', is a refinement of D_i referring to the delay cost incurred for items in bin i over the interval $T_{i'}$.

Let N be the total number of bins opened in a given solution. We define

$$C_B := N\,B, \quad C_D := \sum_{i=1}^{N} D_i, \quad C_H := \sum_{i=1}^{N} H_i,$$

so that the total cost incurred by the solution is equal to

$$C_{tot} := C_B + C_D + C_H.$$

In the special case of the optimal solution, denoted by OPT, we introduce the corresponding terms D_j^*, H_j^* for each opened bin j, and C_B^*, C_H^*, C_D^* for total costs of each type.

3. Release-On-Balance algorithm

We present an algorithm combining PPB with the Any-Fit bin packing scheme [4]. Any-Fit is a class of bin packing algorithms that open a new bin only when the current item does not fit into any of the open but unreleased bins. Items that fit to one of the opened bins can be placed into any of these. Fix any bin packing algorithm $\mathcal{A}$ from the Any-Fit family.

Definition 3.1. *Release-On-Balance (ROB) is an algorithm that uses $\mathcal{A}$ to assign items into bins. For a fixed $\alpha > 0$, each bin i is released exactly when*

$$H_i + D_i = \frac{1}{\alpha} B. \tag{1}$$

The constant $\alpha > 0$ is introduced to obtain the best possible bounds in the chosen analysis framework. We start by proving some useful properties for ROB.

3.1. Properties of ROB

Lemma 3.2. *If ROB has different bins i and i' open at the same time t, then $l_i^t + l_{i'}^t > 1$.*

Corollary 3.3. *If i and i' are different bins used by ROB such that $l_i + l_{i'} \leq 1$, then T_i and $T_{i'}$ are disjoint.*

Lemma 3.4. *Let I be a set of bins in ROB that has $l_i \geq \frac{1}{2}$ for all $i \in I$. Then the number of bins in I satisfies $|I|B \leq 2C_B^*$.*

Proof. The total item load in the instance is at least $\frac{1}{2}|I|$, and since any optimum solution must fit the items to unit size bins the total bin count is $\frac{C_B^*}{B} \geq \frac{1}{2}|I|$. □

Lemma 3.5. *Let $\beta > 0$ and let I be a set of bins in ROB that has $l_i < \frac{1}{2}$ and $D_i \geq \frac{1}{\beta}B$ for all $i \in I$. Then the number of bins in I satisfies $|I|B \leq C_B^* + \beta\, C_D^*$.*

Proof. Fix a bin $i \in I$ and consider the time span T_i from the opening to the release of i. No other bin $i' \in I$, $i' \neq i$, can remain open during T_i due to Lemma 3.2. Thus the intervals T_i, $T_{i'}$ must be disjoint. Further, OPT must either release a bin j during T_i or collect a delay cost of at least $D_i \geq \frac{1}{\beta}B$ during T_i. Considering all $i \in I$ we can write $|I| \leq \frac{C_B^*}{B} + \frac{C_D^*}{\frac{1}{\beta}B}$ and hence $|I|B \leq C_B^* + \beta\, C_D^*$. □

Lemma 3.6. *Let $\beta > 0$ and let I be a set of bins in ROB that has $l_i < \frac{1}{2}$ and satisfies $H_i \geq \frac{1}{\beta}B$ for all $i \in I$. Then the number of bins in I satisfies $|I|B \leq 2C_B^* + \beta\, C_H^*$.*

Proof. No two different bins $i \in I$ and $i' \in I$ can be open at the same time by Lemma 3.2. By definition of I, we further conclude that the time during which ROB keeps the bins in I open is at least $\frac{B}{\beta h}$.

If OPT bin j contains an item placed in a bin $i \in I$ by ROB, we say bin j *overlaps with* bin i. Define

$$I(j) := \{i \in I \mid j \text{ overlaps with } i\}.$$

From the definition of holding costs, we see that an OPT bin j is open for time $\frac{H_j^*}{h}$. Since each of the bins in $I(j)$ is open for at least time $\frac{B}{\beta h}$ but no two of them are open simultaneously, we get the following:

$$|I(j)| \leq 2 + \frac{H_j^*/h}{B/(\beta h)} = 2 + \frac{\beta\, H_j^*}{B}.$$

This is illustrated in Figure 1. Taking a sum over all OPT bins j and observing that $\cup_j I(j) = I$ we get

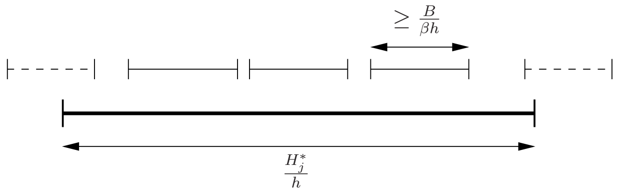

Figure 1: An illustration of the upper bound for the number of ROB bins i overlapping with OPT bin j. The thin segments correspond to each T_i and the thick segment corresponds to T_j on a horizontal time axis.

$$|I|B \leq \sum_j |I(j)|B \leq \sum_j (2B + \beta H_j^*) \leq 2C_B^* + \beta C_H^*.$$

□

3.2. Bounds on ROB competitive ratio

Theorem 3.7. *In the model with both delay and holding costs, ROB with* $\alpha = \frac{5}{2}$ *has competitive ratio at most* 7.

Proof. Because $D_i + H_i = \frac{1}{\alpha}B$ for each ROB bin i, we can partition the bins used by ROB into three sets as follows:

I_1: bins with $l_i \geq \frac{1}{2}$

I_2: bins with $l_i < \frac{1}{2}$ and $D_i \geq \frac{B}{2\alpha}$

I_3: bins with $l_i < \frac{1}{2}$ and $H_i > \frac{B}{2\alpha}$.

We have $C_{tot} = (|I_1| + |I_2| + |I_3|)(B + \frac{B}{\alpha})$. We apply Lemma 3.4 to upper bound $|I_1|$ and Lemmas 3.5 and 3.6 with $\beta = 2\alpha$ to upper bound $|I_2|$ and $|I_3|$, yielding

$$\begin{aligned} C_{tot} &= \left(1 + \frac{1}{\alpha}\right)(|I_1|B + |I_2|B + |I_3|B) \\ &\leq \left(1 + \frac{1}{\alpha}\right)(2C_B^* + C_B^* + 2\alpha C_D^* + 2C_B^* + 2\alpha C_H^*) \\ &= \left(1 + \frac{1}{\alpha}\right)(5C_B^* + 2\alpha(C_D^* + C_H^*)). \end{aligned}$$

Using $\alpha = \frac{5}{2}$ gives $C_{tot} \leq 7(C_B^* + C_D^* + C_H^*) = 7C_{tot}^*$. □

If there are no holding costs, i.e. $h = 0$, a competitive ratio of 4 is attainable.

Theorem 3.8. *When the holding cost rate is zero* ($h = 0$)*, ROB with* $\alpha = 3$ *has competitive ratio at most* 4.

Proof. We split the bins used by ROB into two disjoint sets:

I_1: bins with $l_i \geq \frac{1}{2}$
I_2: bins with $l_i < \frac{1}{2}$.

We have $D_i = \frac{1}{\alpha}B$ for each bin i, so $C_{tot} = (|I_1| + |I_2|)(B + \frac{B}{\alpha})$. We apply Lemma 3.4 to upper bound $|I_1|$ and Lemma 3.5 with $\beta = \alpha$ to upper bound $|I_2|$, yielding

$$\begin{aligned} C_{tot} &= \left(1 + \frac{1}{\alpha}\right)(|I_1|B + |I_2|B) \\ &\leq \left(1 + \frac{1}{\alpha}\right)(2C_B^* + C_B^* + \alpha C_D^*). \end{aligned}$$

Using $\alpha = 3$ gives $C_{tot} \leq \frac{4}{3}(3C_B^* + 3C_D^*) = 4C_{tot}^*$. □

If there are no delay costs, similar techniques yield another bound. We omit the full proof due to a forthcoming improved result in Corollary 4.3.

Theorem 3.9. *If item delay costs are zero, ROB with* $\alpha = 4$ *has competitive ratio at most* 5.

4. Fast release-on-balance algorithm

We now turn to the special case of bilinear cost functions, for which we are able to improve on the bounds for ROB by considering a different algorithm, which is also based on the same idea of balancing delay, holding and release costs.

Definition 4.1. *Fast release-on-balance (FROB) is an algorithm that releases a bin i when either of the conditions is satisfied:*

- *bin i collects holding and delay of* $H_i + D_i = B$
- *the next item does not fit in bin i; a new bin is opened for the item.*

Observe that FROB never has more than one bin open at a time. Before analysing its performance, define

$$(X)^+ := \max(X, 0),$$

whence also $X + (Y - X)^+ = \max(X, Y)$.

Theorem 4.2. *If delay costs are bilinear, FROB has competitive ratio at most 3.*

Proof. Consider the sequence $(1, 2, \ldots, N)$ of bins that are used by FROB for a given instance. We assume bin i is opened and released before bin i' whenever $i < i'$. We divide the bins into a set of bin pairs P and individual bins Q such that each bin is either in Q or a part of exactly one pair in P. Let $c_i = H_i + D_i$. For each bin i with $c_i < B$, we set $(i, i+1)$ as a pair in P and remove bin $i + 1$ from further considerations. The remaining bins all have $c_i = B$ and they belong to Q.

Fix a sequence s of $2M + 1$ consecutive bins used by FROB, with s consisting of $M = M(s)$ pairs from P in the beginning and one bin from Q at the end. Let T_s be the interval between the opening of the first bin that belongs to P and the release of the single type Q bin in the sequence s. This partition of FROB bins, which are opened and released during the time interval T_s, into two sets is illustrated in Figure 2. We prove that during T_s, FROB costs are at most three times the cost of an offline optimal algorithm OPT. Each bin opening cost is considered to occur at the time of release. Making a split of all FROB bins into disjoint subsequences s and summing over s then yields

$$C_{tot} \leq 3C_{tot}^*. \tag{2}$$

Hence, the proof is reduced to the case of a single sequence s.

To make a finer distinction, denote the interval between ROB's opening of the first type P bin and the release of the second bin of the last pair of type P bins by T_P. Denote by T_Q the time interval during which the only type Q bin of s is

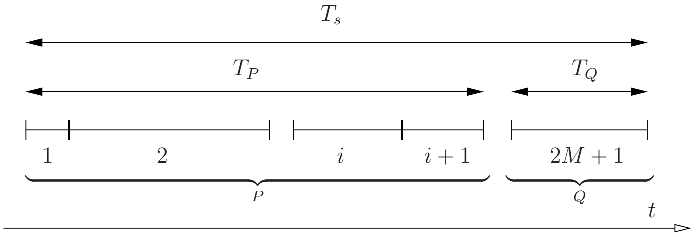

Figure 2: An illustration of the two classes of bins P and Q in a sequence s. FROB released bins 1 and i due to the next item not fitting in the bins, while bin $2M+1$ was released due to time-dependent costs c_i reaching the value B. The bins 2 and $i+1$ may have been released due to either reason.

open. Since for each of the bin pairs in P, the second bin was opened due to the arrival of a new item whose size exceeded the remaining capacity of the first bin, we conclude that FROB is assigning an item load strictly over 1 in each of these pairs. Hence FROB is assigning an item load strictly over M during T_P, and the same holds for OPT.

Assume OPT releases r bins during T_P and r' bins during T_Q, which corresponds to OPT having opening costs of exactly

$$(r+r')B$$

over this time interval. Denote again by T_i the time interval that starts from the arrival of the first item that FROB assigns to bin i and ends with FROB's release of i. Considering the bins that belong to P, at least $2M-r$ of their intervals T_i do not contain an OPT release. Denote the set of these type P bins without OPT release in their interval by P_0, $|P_0| \geq 2M-r$. The lack of OPT releases implies we can charge a combined delay and holding cost of at least c_i to OPT during T_i for each $i \in P_0$. Thus OPT delay and holding costs on T_P are at least

$$\sum_{i\in P_0} c_i.$$

We still need a lower bound on OPT delay and holding costs during T_Q. Let $\bar{l}_Q \leq 1$ be the average load that FROB achieves over T_Q. Formally expressed, $\bar{l}_Q = \frac{1}{|T_Q|}\int_{T_Q} l_Q(t)dt$, where $l_Q(t) \leq 1$ is the instantaneous item load for FROB. Consider first a solution that does not release any bins over the time interval T_s. In this case there would be a load of at least M remaining before the start of T_Q. Thus, a solution without releases in T_s would achieve an average load over T_Q of at least $M+\bar{l}_Q$. Taking into account that in OPT there are $r+r'$ releases during the complete time interval T_s, we conclude the average item load for OPT in T_Q is at least $M+\bar{l}_Q-r-r'$. Hence, OPT collects a delay cost over T_Q of at least

$$\begin{aligned}\frac{(M+\bar{l}_Q-r-r')^+}{\bar{l}_Q}D_Q &= (1+\frac{M-r-r'}{\bar{l}_Q})^+D_Q \\ &\geq (1+M-r-r')^+D_Q.\end{aligned}$$

Consider now OPT holding costs during T_Q. The number of bins that are open during the complete T_Q is equal to at least the average load and hence $\lceil M+\bar{l}_Q-r-r'\rceil$, implying that OPT holding costs are at least $(M+1-r-r')^+H_Q$.

model	any D	bilinear D
$H=0$	4	3
$D=0$	3	
$H,D>0$	7	3

Table 1: Proven upper bounds for FROB and ROB competitive ratios depending on the different costs types. For the case $D=0$ the delay model is trivially bilinear.

By combining the previous results, we thus establish that OPT costs over the interval T_s are at least

$$\begin{aligned}&(r+r')B+\sum_{i\in P_0} c_i+(M+1-r-r')^+D_Q+(M+1-r-r')^+H_Q \\ =&(r+r')B+(M+1-r-r')^+B+\sum_{i\in P_0} c_i \\ =&\max(M+1,r+r')B+\sum_{i\in P_0} c_i.\end{aligned}$$

Algorithm FROB achieves a cost of $2MB+\sum_{i=1}^{2M} c_i+2B$ over time interval T_s. Recall that $c_i \leq B$ due to FROB release rule and that $|P_0| \geq 2M-r$. Thus, the ratio between FROB and OPT costs is

$$\begin{aligned}&\frac{2MB+\sum_{i=1}^{2M} c_i+2B}{\max(M+1,r+r')B+\sum_{i\in P_0} c_i} \\ &\leq \frac{2(M+1)B+\sum_{i\in P_0} c_i}{(M+1)B+\sum_{i\in P_0} c_i}+\frac{\sum_{i\notin P_0,\, i\leq 2M} c_i}{(r+r')B} \\ &\leq 2+\frac{rB}{(r+r')B} \quad \leq 3.\end{aligned}$$

□

The special case of all delay costs being identically zero satisfies bilinearity, and we get the following corollary.

Corollary 4.3. *If there are no delay costs, FROB has competitive ratio at most 3.*

5. Lower bounds

Theorem 5.1. *No deterministic online algorithm can achieve a competitive ratio for the online bin packing problem with delay and holding costs smaller than* 2. *Moreover, this bound holds if either one (but not both) of holding cost rate or delay cost rate is zero and the bin capacity is unlimited.*

Proof. The result is a corollary of Theorem 23 in [5] that applies to generalisations of the TCP acknowledgement problem (without capacities). □

6. Conclusions and further work

In this paper we extend online bin packing by introducing delay and holding costs, which allows the modelling of a wide range of interesting real-world problems. We provide constant-competitive algorithms when bins have uniform capacity and study relevant special cases. Our results are collected in Table 1. Although the lower bounds from the TCP acknowledgement problem carry over in this case, one may ask whether these could be improved upon when there bin capacities and delay or holding costs. We reserve this question for future work. On the other hand, it may be possible to further improve the upper bounds as well, which may be interesting from a practical point of view.

Further, one should note that if the problem involves an additional cost layer, so that there is a second layer the bins need to be released via and bins of the first layer may be packed into bins of the second layer, then the problem with unlimited capacities corresponds to the online version of the well known NP-complete joint-replenishment problem [16], which has applications in sensor-network data aggregation and supply chain optimisation. In the future we want to consider extensions of the ROB scheme to these more complicated cost structures.

Acknowledgement

The first author was supported by Helsinki Graduate School for Computer Science and the Nokia Foundation. The second author was supported by the Cloud Software Programme of the Finnish Strategic Centre for Science, Technology and Innovation TiViT.

References

[1] A. van Vliet, An improved lower bound for on-line bin packing algorithms, Information Processing Letters 43 (1992) 277 – 284.

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

[3] E. G. Coffman, M. R. Garey, D. S. Johnson, Approximation Algorithms for NP-Hard Problems, PWS Publishing Company, pp. 46–93.

[4] G. Galambos, G. J. Woeginger, On-line bin packing - a restricted survey, Mathematical Methods of Operations Research 42 (1995) 25–45.

[5] D. R. Dooly, S. A. Goldman, S. D. Scott, On-line analysis of the TCP acknowledgment delay problem, Journal of the ACM 48 (2001) 243–273.

[6] S. Albers, H. Bals, Dynamic TCP acknowledgement: penalizing long delays, SIAM J. on Discrete Mathematics 19 (2005) 938–951.

[7] J. Frederiksen, K. Larsen, Packet bundling, in: M. Penttonen, E. Schmidt (Eds.), Algorithm Theory - SWAT 2002, volume 2368 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 2002, pp. 1–38.

[8] Karlin, Kenyon, Randall, Dynamic TCP acknowledgment and other stories about e /(e - 1), Algorithmica 36 (2008) 209–224.

[9] H. Fujiwara, K. Iwama, Average-case competitive analyses for ski-rental problems, Algorithmica 42 (2005) 95–107.

[10] H. M. Wagner, T. M. Whitin, Dynamic version of the economic lot size model, Management Science 5 (1958) 89–96.

[11] N. Brahimi, S. Dauzere-Peres, N. M. Najid, A. Nordli, Single item lot sizing problems, European Journal of Operational Research 168 (2006) 1 – 16.

[12] S. Anily, M. Tzur, L. Wolsey, Multi-item lot-sizing with joint set-up costs, Mathematical Programming 119 (2009) 79–94.

[13] U. Wemmerlöv, The part-period balancing algorithm and its look ahead-look back feature: A theoretical and experimental analysis of a single stage lot-sizing procedure, Journal of Operations Management 4 (1983) 23 – 39.

[14] J. Balogh, J. Békési, G. Galambos, New lower bounds for certain classes of bin packing algorithms, in: K. Jansen, R. Solis-Oba (Eds.), Approximation and Online Algorithms, volume 6534 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 2011, pp. 25–36.

[15] J. Y. T. Leung, Handbook of Scheduling: Algorithms, Models, and Performance Analysis, Chapman & Hall/CRC, 1 edition, 2004.

[16] E. Arkin, D. Joneja, R. Roundy, Computational complexity of uncapacitated multi-echelon production planning problems, Operations Research Letters 8 (1989) 61–66.

Appendix A.

In this Appendix we prove that in the bin packing problem with delay and holding costs the strict and asymptotic competitive ratios are equal under only mild assumptions. The strict competitive ratio is the smallest $\alpha \geq 1$ such that $ALG \leq \alpha \, OPT$ for all instances, whereas the asymptotic competitive ratio is the smallest $\alpha \geq 1$ for which there exists a constant c such that $ALG \leq \alpha \, OPT + c$ for all instances.

At first we need a *sanity* assumption on the algorithms: there exists a finite time t_0 such that if no new items arrive in t_0 time units, all bins have been released and there are no open bins anymore. Any algorithm that aims to achieve a good performance should have this property, since otherwise delay or holding cost may grow without limits.

Secondly, we assume the algorithm *resets* its memory buffer every time it releases the last bin. More accurately, if for some input sequence S the last bin is released at a time t, consider S as the concatenation $S = S_1 \circ S_2$ of the item arrival subsequences before and after t. The algorithm is assumed to operate on the S_2 part in exactly the same way as if the input was S_2 alone.

In particular, these assumptions allow us to construct *self-concatenations* S^n consisting of n copies of a base input S, with gaps of length t_0 between consecutive Ss, while maintaining the simple cost equations $OPT(S^n) = n \cdot OPT(S)$, $ALG(S^n) = n \cdot ALG(S)$.

Theorem A1. *Let ALG be a deterministic algorithm for the bin packing problem with delay and/or holding costs. If ALG satisfies the sanity and reset assumptions above, then its strict and asymptotic competitive ratios are equal.*

Proof. The definitions immediately imply that the strict competitive ratio is always at least equal to the asymptotic competitive ratio. Thus, it suffices to prove that lower bounds for the strict competitive ratio are also lower bounds for the asymptotic competitive ratio.

Let β be any (proper) lower bound for the strict competitive ratio of ALG, i.e., let S be a problem instance with

$$\beta \, OPT(S) + \varepsilon = ALG(S)$$

for some $\varepsilon > 0$. We construct the self-concatenation S^n, yielding $OPT(S^n) = n \cdot OPT(S)$, $ALG(S^n) = n \cdot ALG(S)$, and further

$$\beta \, OPT(S^n) + n \cdot \varepsilon = ALG(S^n).$$

This implies that $ALG(S^n)$ can not be bound by $\beta \, OPT(S^n) + c$ for any constant c. Thus any real number β below the strict competitive ratio must be a lower bound for the asymptotic competitive ratio, and the proof is finished. □