# SIGACT News Online Algorithms Column 20: The Power of Harmony

Rob van Stee
Max Planck Institute for Informatics
Saarbrücken, Germany

I would like to begin this column by thanking Marek Chrobak, Lance Fortnow and Brendan Mumey for giving me this opportunity for continuing the online algorithms column started by Marek in 2003. I will begin by discussing a problem close to my heart, namely online bin packing. For the standard version of this problem, there has been very little progress in the last 10 years. I hope to bring some new attention to this important problem and hopefully convince some of you to give this some thought. I will also discuss some (but by no means all) recent results in strip packing and two-dimensional bin packing.

## 1 The classical bin packing problem

Many of you who read this are probably familiar with the Harmonic algorithm [33] for bin packing. I will briefly discuss it and various improvements on it that have been suggested, mainly in order to explain the limitations of this general approach. The best algorithm of this class, Harmonic++, is the best known online algorithm for this problem (and has been for over ten years now). Here we use the asymptotic performance ratio (or asymptotic competitive ratio) as performance measure. That is, we allow an additive constant: an algorithm is $C$-competitive if for all inputs, it has cost at most $C$ times the optimal cost plus some fixed constant.

The Harmonic algorithm is defined as follows. Define *types* for the items based on their sizes: items with size in the interval $(1/(i+1), 1/i]$ are said to be of type $i$ for $i = 1, \ldots, k-1$ (for some value of $k$). For each type, the items are packed in dedicated bins, which contain only items of one specific type. Hence each bin for type $i$ items contains $i$ items of this type ($i = 1, \ldots, k-1$). This leaves items of size at most $1/k$: such items are also packed into dedicated bins, but now using Next Fit [31]. This algorithm has one active (open) bin and packs items into it until some item does not fit. Then, it closes the active bin (and never uses it again) and opens a new bin.

What is the worst thing that can happen for this algorithm from the viewpoint of competitive analysis? We need to find an input for which a very efficient packing exists, but which HARMONIC packs badly. The first thing that can be seen is that w.l.o.g., all items in the input will be of the form $1/i + \varepsilon$ (for different values of $i > 1$, and some very small value of $\varepsilon$). This ensures that the amount of unused space in each bin of HARMONIC is maximized. Furthermore, it minimizes the amount of space that is needed in the optimal solution to pack these items. As it turns out, the worst case input is given by a sequence of items that is defined as follows. Start with an item of type 1 and size $1/2 + \varepsilon$. Now, in each step, add an item of the largest possible type that can fit with all previous items into a single bin. We can repeat this until the next item will be of type $k$. It can be shown that the sizes in this input sequence are given by $1/t_i + \varepsilon$ $(i \geq 1)$, where $t_i$ is defined by

$$t_1 = 2, \quad t_{i+1} = t_i(t_i - 1) + 1 \quad i \geq 1.$$

The first few numbers of this sequence are $2, 3, 7, 43, 1807$. This sequence was first examined by Sylvester [39].

Since we allow an additive constant to the competitive ratio, in order to turn the above set of items into a real lower bound input for any online algorithm, we need to repeat each item in the input $N$ times for some arbitrarily large $N$. To summarize the preceding discussion, the input has the following form:

$$N \times \left(\frac{1}{2} + \varepsilon\right), N \times \left(\frac{1}{3} + \varepsilon\right), N \times \left(\frac{1}{7} + \varepsilon\right), N \times \left(\frac{1}{43} + \varepsilon\right), N \times \left(\frac{1}{1807} + \varepsilon\right), \dots \qquad (1)$$

where the items are given to the algorithm in order of nondecreasing size. For large $k$, the resulting lower bound on the competitive ratio then tends to

$$h_\infty = \sum_{i=1}^{\infty} \frac{1}{t_i - 1} \approx 1.69103.$$

**Improvements over** HARMONIC No bounded space algorithm (i.e., an algorithm which has only a constant number of bins open at any time) can do better than HARMONIC. This holds because all bounded space algorithms need to pack almost all items of the input (1) into dedicated bins that contain only items of one type, that is, exactly the way HARMONIC packs them. For instance, after all items of size $\frac{1}{1807} + \varepsilon$ have arrived, all but a constant number of bins used so far are already closed, so almost all items of the next size $\frac{1}{43} + \varepsilon$ have to be packed into new bins, etc.

However, if we allow unbounded space, it is possible to improve. Looking at the input sequence (1), it can be seen that HARMONIC wastes the most space in the bins containing only items of size $1/2 + \varepsilon$. It would be much better to combine these items with other input items into the same bins. To do this in a structured way, we can divide the intervals $(1/2, 1]$ and $(1/3, 1/2]$ into two parts, so that two items of the lower halves of both intervals can definitely be packed together into a single bin. That is, we have the intervals $(1/3, y], (y, 1/2], (1/2, 1 - y], (1 - y, 1]$ for some $1/3 < y < 1/2$. Then we reserve some fraction of the bins that are used to pack items of size in $(1/3, y]$ for future items of size in $(1/2, 1 - y]$ (hoping that such items will indeed arrive). Hence some bins will contain two items of size in $(1/3, y]$, and some only 1. Items of sizes in $(y, 1/2]$ and $(1 - y, 1]$ are packed as before. Lee and Lee [33] used this approach with $y = 37/96$ to get a competitive ratio of $373/228 \approx 1.6359$.

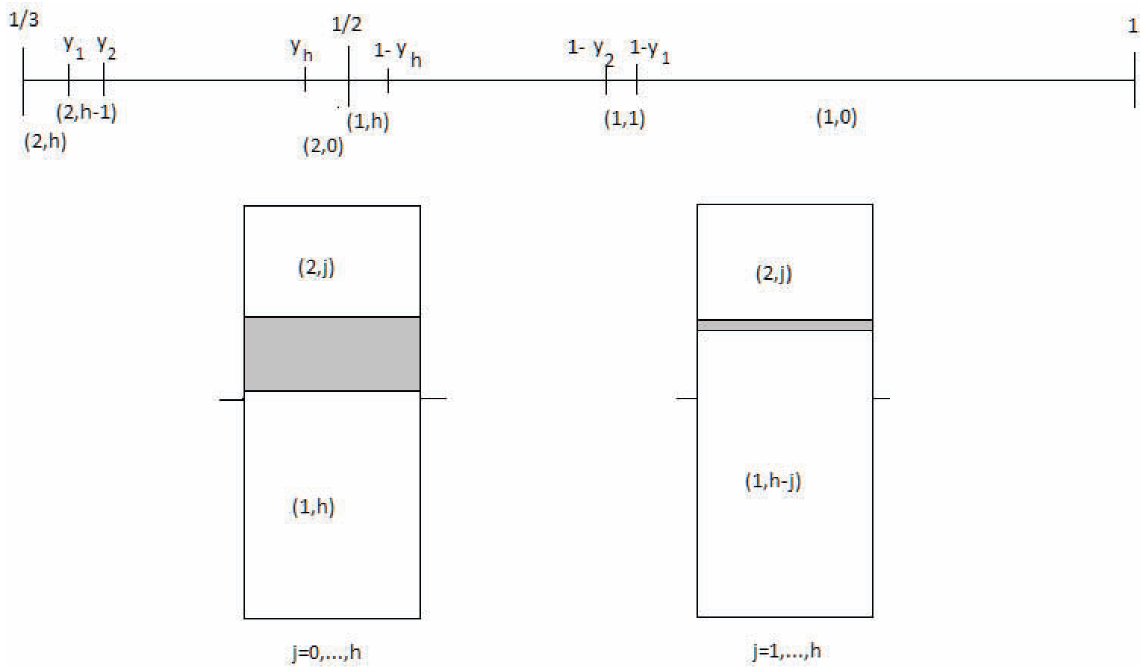

Figure 1: Breakpoints used by Harmonic-type algorithms, and the $2h + 1$ lower bound inputs. Gray areas indicate sand (tiny items).

Of course, once we have this idea of subdividing the intervals $(1/2, 1]$ and $(1/3, 1/2]$, we can use more subintervals. Seiden [36] perfected this approach using over 70 intervals to give an algorithm called Harmonic++ with competitive ratio 1.58889. This is about as far as we can get with this type of algorithm: Ramanan et al. [35] showed already back in 1989 that this type of algorithm cannot do better than $19/12 \approx 1.58333$. To be precise, they considered algorithms which satisfy the following rules.

Define a set of values $1/3 = y_0 < y_1 < \ldots < y_h < y_{h+1} = 1/2$. For $i = 1, \ldots, h$, an item is of type $(1, i)$ if it has size in $(1 - y_{i+1}, 1 - y_i)$, and it is of type $(2, i)$ if it has size in $(y_{h-i}, y_{h-i+1})$. In addition, type $(1, 0)$ is the interval $(1 - y_1, 1)$, and type $(2, 0)$ is the interval $(y_h, 1/2)$. Finally, define a value $\lambda \leq 1/3$. Items are of type $\lambda$ if they have size at most $\lambda$. The lower bound then holds for all algorithms which behave as follows.

1. The number of bins with one item of type $(2, j)$ is a fixed fraction $1/m_{2,j}$ of the total number of items of this type in the input.

2. Bins contain only combinations of types that are guaranteed to fit for any set of items of these types. That is, no bin contains type $(1, i)$ and type $(2, j)$ for $i + j \leq h, i \geq 0, j \geq 0$.

3. No bin contains an item of type $\lambda$ together with any of the other defined types.

Note that there are no restrictions on how items with size in the interval $(\lambda, 1/3]$ are packed; they may even be combined with other types in one bin.

For such algorithms, we consider $2h + 1$ possible inputs, shown in Figure 1. In all cases, there will be one item of some type $(1, i)$, one $(2, j)$ item, and many very small items (sand) of type $\lambda$

to fill up the bin. Each item mentioned in the previous sentence actually arrives $N$ times for some large $N$, hence for every list, we have that the optimal cost is $N$ for every input. Since the algorithm must deal with all of these inputs using the same fixed fractions $1/m_{2,j}$, the lower bound follows. (We get $2h+1$ inequalities for the $2h+2$ variables $\{y_i\}_{i=1}^h, \{m_{2,j}\}_{j=1}^h, \lambda$, and the competitive ratio. We minimize the competitive ratio so that all inequalities are satisfied.)

It seems very difficult to improve on this type of algorithm. It is tempting to e.g. try and combine items together whenever they fit. For instance, if we have an item of size $1-y_j+\varepsilon$ and an item of size $y_j-\varepsilon$ for some $j \in \{1,\dots,h\}$ and very small $\varepsilon$, we would like to be able to combine them in one bin. But any algorithm as described above will fail to do this. If we change the algorithm to allow for these kind of combinations, it means that the output is no longer based purely on the type of an item but also on its exact size. Then the analysis framework which was used before will no longer work, since it analyzes inputs based only on the types of the items in it.

Another idea would be to try and combine some large types of items (e.g. type $(1,1)$, or a newly defined subtype of $(1,0)$) with the smallest type of items. Currently, the smallest type is packed completely separate from the rest of the input. This means that the bins containing these items are almost completely full, but if large items arrive later, it could be better to anticipate them and keep some bins with a few small items ready. This idea was used in an online bin packing algorithm that uses resource augmentation [20], but it is unclear whether it can also help in the standard setting. In any event, by the lower bound construction of Ramanan et al. [35] it seems to be more important to pack relatively large items well.

**General lower bounds** How well can unbounded space algorithms do in general? Yao showed that no online algorithm has performance ratio less than $\frac{3}{2}$ [41]. Brown and Liang independently improved this lower bound to 1.53635 [9, 34], using the sequence (1). We now no longer have the property that items of different sizes in this sequence are necessarily packed into different bins, because the algorithm may keep many bins open for future items. Still this kind of sequence appears to give the best lower bounds. Van Vliet showed how to use the input defined by (1) to prove a lower bound of 1.54014. Van Vliet set up a linear programming formulation to define all possible online algorithms for this input to prove the lower bound.

This input appeared to be "optimal" to make life hard for online algorithms: the smallest items arrive first, and the input is constructed in such a way that each item is as large as possible given the larger items (that are defined first). Surprisingly however, in 2010 Balogh et al. [4] managed to prove a lower bound of $248/161 \approx 1.54037$ using a slight modification of the input. Instead of using 1/43 as the fourth item size, they use 1/49. We then get the following input:

$$N \times \left(\frac{1}{2}+\varepsilon\right), N \times \left(\frac{1}{3}+\varepsilon\right), N \times \left(\frac{1}{7}+\varepsilon\right), N \times \left(\frac{1}{49}+\varepsilon\right), N \times \left(\frac{1}{343}+\varepsilon\right), \dots \qquad (2)$$

For the input that consists only of the first four phases of this input, the resulting lower bound is now slightly lower than before, but this is more than compensated by the next items. Other variations of this input sequence do not seem to give higher bounds. Can we do better, or perhaps show a lower bound for a wider class of instances than Ramanan et al. [35] considered?

All lower bound constructions that we have seen are of a very specific type. There are a few different sizes in the input, each item arrives $N$ times for some large $N$. Items typically arrive in order from small to large and the only uncertainty that the online algorithm is faced with is the question of when the input ends. Basically, the online algorithm "knows" the entire input in advance, just not how much of it will actually arrive.

**More on bounded space algorithms** A problem with the lower bound analysis for bounded space algorithms above is that it also holds for offline bounded space algorithms and for algorithms that use exponential time. There is no good packing for this sequence which can be achieved using bounded space. It is natural to instead compare the performance of online bounded space algorithms to the best possible offline *bounded space* algorithm.

Chrobak et al. [12] defined the $k$-bounded-space ratio, which compares an online algorithm $A$ to a $k$-bounded space offline algorithm $\textsc{opt}_k$, that has at most $k$ bins open at any time. The authors show that for this measure, it is not possible to give a 2-bounded PTAS, or even a 2-bounded approximation algorithm with approximation ratio below 5/4. They give a 2-bounded offline algorithm with approximation ratio $3/2 + \varepsilon$, and also showed a surprising result that it is NP-hard to distinguish between inputs that require 4 bins to pack and those that require 6 bins to pack in the optimal 2-bounded solution.

For online 2-bounded space algorithms, they showed a lower bound on this ratio of 4/3. They also mention that Epstein and Levin afterwards improved this lower bound to 3/2. Since Csirik and Johnson showed that the version of Best Fit which uses only two open bins has asymptotic worst case ratio of 1.7 [14], this shows that its 2-bounded space ratio is in the interval $[1.5, 1.7]$, since $\textsc{opt}_k(I) \geq \textsc{opt}(I)$ for any input $I$. Decreasing the gap here is an interesting open problem.

Best Fit places each new item into the fullest bin which can accomodate it. In a forthcoming paper, Sgall [38] analyzed a class of algorithms he calls Good Fit, which pack items according to the following rules:

1. Pack the item into some bin which is more than 1/2 full;

2. If there is no such bin, pack it into some bin where it fits;

3. Else, open a new bin.

Additionally, these algorithms close bins according to the following rules. If there are at least three open bins, the algorithm *can* close a bin except the one possibly just opened for a new item that satisfies

1. The bin is at least 5/6 full;

2. If there is no such bin, the bin to be closed must be at least 2/3 full;

3. If there is no such bin either, a bin must be at least 1/2 full to be closed.

The Best Fit algorithm never closes any bin and always uses the fullest bin possible to pack an item, thus it is a Good Fit algorithm. Sgall [38] shows that all Good Fit algorithms (including $k$-bounded space ones) use at most $\lfloor 1.7 \cdot \textsc{opt}(I) + 0.7 \rfloor \leq \lceil 1.7 \cdot \textsc{opt}(I) \rceil$ bins, improving and generalizing the upper bound of $\lceil 1.7 \cdot \textsc{opt}(I) \rceil$ by Garey et al. [22].

## 1.1 Variations

**Dynamic bin packing** It is natural to consider the version of the problem where items may also depart. In this case, as we saw above with bounded space algorithms, there are two possible optimal solutions that we can compare the performance of an online algorithm with: one that is allowed to repack items and can give the globally optimal solution at any time, and one that like the online algorithm cannot repack. Most results are for the version where the optimal solution can repack items.

This problem was first considered by Coffman, Garey and Johnson [13]. They gave an upper bound of 2.897 for First Fit and a tight upper bound of 2.788 for Modified First Fit, as well as a lower bound of 2.388. This lower bound was improved to 2.428 by Chan et al. [10] and then to 2.5 by Chan et al. [11]. This last lower bound even holds if the offline optimal algorithm is not allowed to repack items.

A special case of interest for this problem is that of unit fraction items, where each item has size of the form $1/i$ for some integer $i$. This problem was introduced by Bar-Noy et al. [6] as a generalized version of windows scheduling (WS): given a sequence of $n$ positive integers $w_1, \ldots, w_n$ called windows, that are associated with $n$ equal-length information pages (requests), the goal is to schedule all the pages on broadcasting channels such that the gap between two consecutive appearances of page $i$ on the channel is at most $w_i$ slots, where a slot is the time to broadcast one page.

WS is not the same as unit fraction bin packing (UFBP) as can be seen from the input $\{2, 3, 6\}$. The three items $1/2, 1/3, 1/6$ can be packed together in one bin, but there is no schedule of these pages for WS that uses only one channel. In general, windows scheduling without migration is therefore a restricted version of UFBP. UFBP can be seen as a fractional version of WS that measures the power of unlimited preemptions: it demonstrates what can be achieved when a request is not necessarily transmitted nonpreemptively in one slot, but can instead be partioned into small segments.

Chan et al. [10] showed that First Fit has a competitive ratio at most 2.4942 and at least 2.45 for unit fraction bin packing.

Since items may depart, it seems reasonable to allow the online algorithm to repack some items. This is the *fully dynamic bin packing problem* introduced by Ivkovič and Lloyd [29], who showed a lower bound of 4/3 for any online algorithm that can only repack a constant number of items per step. This lower bound was improved to 1.3871 by Balogh et al. [5].

**Bin packing with rejection** In many applications, it is possible to refuse to pack an item. This rejection needs to be compensated, and costs some given amount for each item, which is called the rejection cost (or rejection penalty) of the item. In an application where bins are disks and items are files to be saved on these disks, the rejection cost of a file is the cost of transferring it to be saved on alternative media. In another application, where bins are storage units, a rejection cost is paid to a disappointed customer whose goods cannot be stored.

Dosa and He [16] were the first to consider the problem of minimizing the sum of rejection costs of rejected items and bins needed to pack accepted items. Epstein [18] showed that the Harmonic algorithm can be extended to deal with rejections while still maintaining the same asymptotic competitive ratio, implying that it remains optimal. She also gave an extension of Modified Harmonic [35] with a ratio of 1.61562, and an asymptotic polynomial time approximation scheme for the offline problem.

**Cardinality constraints** The classical bin packing problem assumes no limit on the number of items which may be packed into a single bin. In practice, many applications require such a bound either due to overheads or additional constraints that are not modeled. For example, a disk cannot keep more than a certain number of files, even if these files are indeed very small. A processor cannot run more than a given number of tasks during a given time, even if all tasks are very short. The problem where there is a given bound $k > 1$ on the number of items which can coexist in one bin is called bin packing with cardinality constraints.

Epstein [17] showed that the HARMONIC algorithm can also be extended to deal with cardinality constraints. Her extension achieves the best possible competitive ratio for bounded space algorithms for any $k > 1$. This competitive ratio is growing as a function of $k$ and surprisingly tends to $1 + h_\infty \approx 2.69103$ (instead of $h_\infty$ as one might expect as $k$ tends to infinity). She also gave optimal algorithms (for any $k$) for variable-sized and resource augmented bin packing with cardinality constraints. Note that the competitive ratio of HARMONIC itself only tends to $h_\infty$ from above as the number of used classes grows, and HARMONIC is not optimal for small numbers of open bins.

**Resource augmentation** The standard competitive analysis sometimes gives results that are too negative. One way around this that has been suggested is to use resource augmentation: give the online algorithm more power (in this case, larger bins) than the offline algorithm that it is compared to. Both bounded space [15] and unbounded space [20] algorithms have been considered in this framework. Boyar et al. [8] analyzed NEXT FIT and WORST FIT with resource augmentation and showed that WORST FIT has a strictly better asymptotic competitive ratio than NEXT FIT for any online bin size in the open interval (1,2), whereas they are the same for larger bin sizes and for bins of size 1. This supports the intuition that WORST FIT should be better than NEXT FIT, since WORST FIT never opens a new bin if an existing bin can still be used. However, both algorithms have an asymptotic competitive ratio of 2.

## 2 Strip packing

In this problem, the items are rectangles, and we are given a strip of width 1 and unbounded height. The goal is to pack all the items without overlap and minimize the maximum height used by any item. This problem is closely related to job scheduling on parallel machines in the setting where each job requires some fixed number of parallel machines to be processed. The difference is that in scheduling, it usually does not matter which machines are used, whereas in strip packing, items must be placed in the strip in one piece.

Han et al. [24] showed that strip packing is essentially the same as (one-dimensional) bin packing in the sense that we can achieve exactly the same results, both offline and online. However, this is only true if we use the *asymptotic* performance ratio as our performance measure. For the absolute competitive ratio (i.e., we do not allow an additive constant), Hurink and Paulus [28] and Ye et al. [42] independently showed an upper bound of $7/2+\sqrt{10} \approx 6.6623$. This improves and generalizes a previous result of Baker and Schwarz [3], who had the additional restriction that all items have height at most 1. Both algorithms are *shelf algorithms*. A shelf algorithm uses a one-dimensional bin packing algorithm $A$ and a parameter $\alpha \in (0, 1)$. Items are classified by height: an item is in class $s$ if its height is in the interval $(\alpha^{s-1}, \alpha^s]$. Each class is packed in separate *shelves*, where we use $A$ to fill a shelf and open a new shelf when necessary. Note that the algorithm $A$ is not necessarily on-line. See Figure 2 for an illustration of a shelf algorithm.

The new algorithm is fairly straightforward: it packs items of width more than 1/2 in dedicated shelves, and packs other items using FIRST FIT into shelves with geometrically rounded heights. Can we do better, for instance by not using shelves?

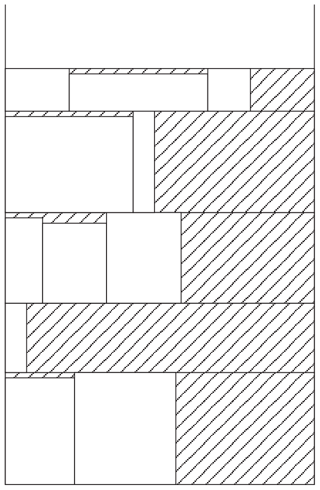

Figure 2: A packing using shelves

Brown, Baker and Katseff [2] showed a lower bound of 2 for the absolute competitive ratio of any online algorithm. Their idea was to construct an alternating sequence of thin and blocking items. Thin items have negligible width and can all be packed next to eachother in an optimal solution. Blocking items have width 1. The height of each item is chosen in such a way that it must be packed above all previous items. For the thin items, this means that each new height is more than the distance between any two previous blocking items. For blocking items, the heights are such that no blocking item fits between a previous blocking item and a thin item.

The analysis of the lower bound implied by this input sequence was improved to 2.25 by Johannes [30], 2.43 by Hurink and Paulus [27], and finally to 2.457 by Kern and Paulus [32], who also showed that there is an algorithm with a matching competitive ratio for this input. This algorithm creates no gaps in the packing except under the first thin item and above the second thin item. These gaps are chosen to be as large as possible while remaining 2.457-competitive.

Harren and Kern [26] modified the input of Brown et al. [2] by releasing slowly growing thin items instead of one thin item between two blocking items. The thin items start at the previous height and grow until they have the next height of the above input. Thus, the online algorithm can choose to place some thin items next to previous thin items, but the effect of this is that the gaps above and below these thin items become smaller, so that in the end the next blocking item does not have to be quite as high as in the original construction, increasing the implied lower bound. Using this construction, they proved that no online algorithm can be better than 2.589-competitive, and they gave an online algorithm for packing this type of input (consisting only of thin and blocking items) with competitive ratio $(3+\sqrt{5})/2 \approx 2.618$. Hence, improving the lower bound significantly will require new ideas; to my mind, the upper bound should have more room for improvement.

**A variation: items appear from the top** A "Tetris like" online model was studied in a few papers. This is similar to strip packing, however, in this model, a rectangle cannot be placed directly in its designated area, but it arrives from the top as in the Tetris game, and should be moved continuously around only in the free space until it reaches its place, (see figure 3), and then cannot be moved again.

In [1], the model was introduced by Azar and Epstein. In that paper, both the rotatable and the

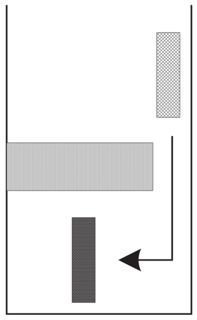
Figure 3: The process of packing an item in the "Tetris like" model

oriented models were studied. For the rotatable model, a 4-approximation algorithm was designed. For the special case where the input consists only of squares, this was improved to $34/13 \approx 2.6154$ by Fekete et al. [21] using a slot-based algorithm. This algorithm divides the strip into one slot of width 1, two slots of width 1/2, four slots of width 1/4 etc. For each item, we first round up its size to the first (negative) power of 2. Then, the item is placed greedily into the slot where it can be placed the lowest by moving it down along the left boundary of the slot until another square or the bottom of the strip is reached.

The situation for the oriented problem is more difficult, as no algorithm with constant approximation ratio exists for unrestricted inputs. If the width of all items is bounded below by $\epsilon$ and/or bounded above by $1-\epsilon$, Azar and Epstein [1] showed a lower bound of $\Omega(\sqrt{\log \frac{1}{\epsilon}})$ on the performance ratio of any online algorithm for any deterministic or randomized algorithm. Restricting the width, they designed an $O(\log \frac{1}{\epsilon})$-approximation algorithm.

## 3 Two-dimensional bin packing

The general approach for this problem has been the following. Use two one-dimensional bin packing algorithms $A$ and $B$ as subtroutines. Use algorithm $A$ to allocate items to *slices* of height 1 and width determined by the type of the width of the item. ($A$ decides which slice to use for any given item, and when to open a new slice.) Use algorithm $B$ to allocate slices to bins based on the type of their width. We call the resulting algorithm $A \times B$. It can be slightly improved by flipping a coin at the start of execution and choosing with equal probability between $A \times B$ and $B \times A$. This algorithm is denoted by $A \otimes B$.

Ideally, we would like to use the algorithm Harmonic++ both as algorithm $A$ and algorithm $B$, because it is the best one-dimensional bin packing algorithm that we know. Hopefully, the competitive ratio of Harmonic++$\otimes$ Harmonic++would be $1.58889^2 = 2.5245$. However, it is not at all clear how to analyze this algorithm and prove this upper bound. Indeed, the best algorithm that was analyzed earlier [37] uses Harmonic in one dimension and Improved Harmonic in the other. Recently, Han et al. [23] managed to analyze the algorithm Harmonic$\otimes SH+$, where $SH+$ is an instance of a Super Harmonic algorithm, using a clever modification of the previous technique to upper bound the maximum weight that can be packed into a single bin. With this they improved the upper bound from 2.66013 to 2.5545.

Note that the best known lower bound for this problem is only 1.907 by Blitz, van Vliet, Woeginger [7], which is an unpublished (and now lost [40]) manuscript. This should be an open invitation to anybody to recreate or improve this bound, as well as the bounds for higher dimensions from the same manuscript! The gap between the upper and lower bounds remains relatively large to this day, and it is unclear how to improve either of them significantly. Note that in particular, it should most likely be possible to prove lower bounds for HARMONIC-based algorithms as described above (that use versions of HARMONIC in both dimensions) which are significantly higher than the general lower bounds, mirroring the situation in one dimension.

An interesting special case is where all items are squares (or cubes). Han et al. [25] presented another SUPER HARMONIC algorithm to improve the upper bounds for online square (2.1187) and cube (2.6161) packing. For these problems, previously IMPROVED HARMONIC algorithms had been used [19]. The lower bounds for square and cube packing remain at 1.6406 and 1.6680 [19]. Can we improve them, for instance for HARMONIC-like algorithms?

## References

[1] Yossi Azar and Leah Epstein. On two dimensional packing. *J. Alg.*, 25(2):290–310, 1997.

[2] Brenda S. Baker, Donna J. Brown, and Howard P. Katseff. Lower bounds for two-dimensional packing algorithms. *Acta Inform.*, 8:207–225, 1982.

[3] Brenda S. Baker and J. S. Schwartz. Shelf algorithms for two-dimensional packing problems. *SIAM J. Comput.*, 12:508–525, 1983.

[4] János Balogh, József Békési, and Gábor Galambos. New lower bounds for certain classes of bin packing algorithms. In Klaus Jansen and Roberto Solis-Oba, editors, *Approximation and Online Algorithms - 8th International Workshop, WAOA 2010*, volume 6534 of *LNCS*, pages 25–36. Springer, 2011.

[5] János Balogh, József Békési, Gábor Galambos, and Gerhard Reinelt. Lower bound for the online bin packing problem with restricted repacking. *SIAM J. Comput.*, 38(1):398–410, 2008.

[6] Amotz Bar-Noy, Richard E. Ladner, and Tami Tamir. Windows scheduling as a restricted version of bin packing. *ACM Trans. Alg.*, 3(3), 2007.

[7] David Blitz, Andre van Vliet, and Gerhard J. Woeginger. Lower bounds on the asymptotic worst-case ratio of online bin packing algorithms. Unpublished manuscript, 1996.

[8] Joan Boyar, Leah Epstein, and Asaf Levin. Tight results for Next Fit and Worst Fit with resource augmentation. *Theor. Comput. Sci.*, 411(26-28):2572–2580, 2010.

[9] Donna J. Brown. A lower bound for on-line one-dimensional bin packing algorithms. Technical Report R-864, Coordinated Sci. Lab., Urbana, Illinois, 1979.

[10] Joseph Wun-Tat Chan, Tak Wah Lam, and Prudence W. H. Wong. Dynamic bin packing of unit fractions items. *Theor. Comput. Sci.*, 409(3):521–529, 2008.

[11] Joseph Wun-Tat Chan, Prudence W. H. Wong, and Fencol C. C. Yung. On dynamic bin packing: An improved lower bound and resource augmentation analysis. *Algorithmica*, 53(2):172–206, 2009.

[12] Marek Chrobak, Jiří Sgall, and Gerhard J. Woeginger. Two-bounded-space bin packing revisited. In Camil Demetrescu and Magnús M. Halldórsson, editors, *Algorithms - ESA 2011 - 19th Annual European Symposium*, volume 6942 of *Lecture Notes in Computer Science*, pages 263–274. Springer, 2011.

[13] Edward G. Coffman, Michael R. Garey, and David S. Johnson. Dynamic bin packing. *SIAM J. Comput.*, 12:227–258, 1983.

[14] János Csirik and David S. Johnson. Bounded space on-line bin packing: Best is better than First. *Algorithmica*, 31(2):115–138, 2001.

[15] János Csirik and Gerhard J. Woeginger. Resource augmentation for online bounded space bin packing. *J. Algorithms* 44(2):308–320, 2002.

[16] György Dósa and Yong He. Bin packing problems with rejection penalties and their dual problems. *Inf. Comput.*, 204(5):795–815, 2006.

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

[18] Leah Epstein. Bin packing with rejection revisited. *Algorithmica*, 56(4):505–528, 2010.

[19] Leah Epstein and Rob van Stee. Online square and cube packing. *Acta Inform.*, 41(9):595–606, 2005.

[20] Leah Epstein and Rob van Stee. Online bin packing with resource augmentation. *Discr. Optim.*, 4(3-4):322–333, 2007.

[21] Sándor Fekete, Tom Kamphans, and Nils Schweer. Online square packing. In Frank Dehne, Marina Gavrilova, Jörg-Rüdiger Sack, and Csaba Tóth, editors, *Algorithms and Data Structures (WADS 2009)*, volume 5664 of *LNCS*, pages 302–314. Springer Berlin / Heidelberg, 2009.

[22] Michael R. Garey, Ronald L. Graham, David S. Johnson, and Andrew Chi-Chi Yao. Resource Constrained Scheduling as Generalized Bin Packing. *J. Combin. Theory Ser. A* 21:257–298, 1976.

[23] Xin Han, Francis Y. L. Chin, Hing-Fung Ting, Guochuan Zhang, and Yong Zhang. A new upper bound 2.5545 on 2d online bin packing. *ACM Trans. Alg.*, 7(4):50, 2011.

[24] Xin Han, Kazuo Iwama, Deshi Ye, and Guochuan Zhang. Strip packing vs. bin packing. In Ming-Yang Kao and Xiang-Yang Li, editors, *Algorithmic Aspects in Information and Management, Third International Conference, AAIM 2007*, volume 4508 of *LNCS*, pages 358–367. Springer, 2007.

[25] Xin Han, Deshi Ye, and Yong Zhou. A note on online hypercube packing. *Central European Journal of Operations Research*, 18:221–239, 2010.

[26] Rolf Harren and Walter Kern. Improved lower bound for online strip packing - (extended abstract). In Roberto Solis-Oba and Giuseppe Persiano, editors, *Approximation and Online Algorithms - 8th International Workshop, WAOA 2010*, volume 7164 of *LNCS*, pages 211–218. Springer, 2011.

[27] Johann Hurink and Jacob Jan Paulus. Online scheduling of parallel jobs on two machines is 2-competitive. *Operations Research Letters*, 36:51–56, 2008.

[28] Johann Hurink and Jacob Jan Paulus. Improved online algorithms for parallel job scheduling and strip packing. *Theor. Comput. Sci.*, 412(7):583–593, 2011.

[29] Zoran Ivkovic and Errol L. Lloyd. A fundamental restriction on fully dynamic maintenance of bin packing. *Inf. Process. Lett.*, 59(4):229–232, 1996.

[30] Berit Johannes. Scheduling parallel jobs to minimize the makespan. *Journal of Scheduling*, 9:433–452, 2006.

[31] David S. Johnson. Fast algorithms for bin packing. *J. Comput. Systems Sci.*, 8:272–314, 1974.

[32] Walter Kern and Jacob Jan Paulus. A tight analysis of Brown-Baker-Katseff sequences for online strip packing. In Ulrich Faigle, Rainer Schrader, and Daniel Herrmann, editors, *Cologne-Twente Workshop (CTW 2010)*, pages 109–110, 2010.

[33] C. C. Lee and D. T. Lee. A simple online bin packing algorithm. *J. ACM*, 32:562–572, 1985.

[34] Frank M. Liang. A lower bound for online bin packing. *Inform. Process. Lett.*, 10:76–79, 1980.

[35] Prakash V. Ramanan, Donna J. Brown, C. C. Lee, and D. T. Lee. Online bin packing in linear time. *J. Alg.*, 10:305–326, 1989.

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

[37] Steve S. Seiden and Rob van Stee. New bounds for multi-dimensional packing. *Algorithmica*, 36(3):261–293, 2003.

[38] Jiří Sgall. A new analysis of Best Fit bin packing. In *Fun with Algorithms, 6th International Conference (FUN 2012)*, 2012. To appear.

[39] James J. Sylvester. On a point in the theory of vulgar fractions. *Amer. J. Math.*, 3:332–335, 1880.

[40] Gerhard J. Woeginger. Personal communication, 2011.

[41] Andrew Chi-Chi Yao. New algorithms for bin packing. *J. ACM*, 27:207–227, 1980.

[42] Deshi Ye, Xin Han, and Guochuan Zhang. A note on online strip packing. *J. Comb. Optim.*, 17(4):417–423, 2009.