Improved Results on Online Dynamic Bin Packing *

Mihai Burcea (Speaker) †‡ Prudence W.H. Wong † Fencol C.C. Yung †

1 Introduction

Bin packing is an NP-hard [10] classical combinatorial optimization problem. The objective is to pack a set of items into a minimum number of unit-size bins such that the total size of the items in a bin does not exceed the bin capacity. The problem has been studied extensively both in the offline and online settings [7, 8, 5]. In the online setting [11, 12], items may arrive at arbitrary time; item arrival time and item size are only known when an item arrives.

Online dynamic bin packing. Most existing work focuses on "static" bin packing in the sense that items do not depart. In some potential applications like warehouse storage, a more realistic model takes into consideration of dynamic arrival and departures of items. In dynamic bin packing (DBP) [6] items arrive over time, reside for some period of time, and may depart at arbitrary time. Each item has to be assigned to a bin from the time it arrives until it departs. The objective is to minimize the maximum number of bins used over all time. In the online setting, the size and arrival time is only known when an item arrives and the departure time is only known when the item departs.

Previous work also focuses on item size being an arbitrary real number in $(0, 1]$ that must be packed into unit-sized bins. We refer to this type of item as a *general size* item. However, in some applications item size is not represented by general size items. Bar-Noy et al. [2] initiated the study of the *unit fraction bin packing problem*, a restricted version where all sizes of items are of the form $\frac{1}{k}$, for some integer k. The problem was motivated by the window scheduling problem [1, 2].

Our contributions. Our main contribution is to improve the lower bound of 1-dimensional online dynamic bin packing of general size items for any deterministic online algorithm from 2.5 [4] to $8/3 \sim 2.666$. Furthermore, we study 2- and 3-dimensional online dynamic bin packing of unit fraction (UF) items. We obtain competitive ratios of 6.7850 and 21.6108 for 2-D and 3-D UF items, respectively. These are in contrast to upper bounds of 7.788 and 22.788 obtained in [13] for 2-D and 3-D general size items, respectively.

Notations and definitions. We denote by *s-item* a 1-D item of size s. For 2-D items, we use the notation $\mathrm{T}(w, h)$ to refer to the type of items with width w and height h. We use '$*$' to mean that the length can take any value at most 1, e.g., $\mathrm{T}(*, *)$ refers to all

*The results appear in ISAAC 2012 and CIAC 2013.

†{m.burcea,pwong}@liverpool.ac.uk, ccyung@graduate.hku.hk. Department of Computer Science, University of Liverpool, Ashton Building, Ashton Street, Liverpool, L69 3BX, UK.

‡Supported by EPSRC Studentship.

items. The parameters w (and h) may take an expression $\leq x$ meaning that the width is at most x. In a similar way we use the notation $\mathrm{T}(x_1, x_2, x_3)$ for 3-D items.

When packing a new item to a bin, repacking of existing items within the same bin is permitted, however migration to other bins and rotation of items are not allowed.

2 An 8/3 Lower Bound for 1-D Online DBP

We consider the 1-D online dynamic bin packing problem of general size items, where each item is represented by a real number in $(0, 1]$. We improve the lower bound from 2.5 [4] (previously 2.388 [6] and 2.428 [3]) to $8/3 \sim 2.666$ (Theorem 1). The improvement stems from an adversarial sequence that forces an online algorithm $\mathcal{A}$ to open $2s$ bins with items having a total size of s only.

Suppose we have s bins each with an $\frac{1}{3}$-item. If we want to force $\mathcal{A}$ to open s new bins using items of size $\frac{2}{3}$, we may need to release $2s$ such items because each existing bin can pack one item. The maximum load in total including the $\frac{1}{3}$-items would be $s(1 + \frac{2}{3})$. On the other hand, if the original s bins have items of size from $\frac{1}{3}$, $\frac{1}{3}+\delta$, $\frac{1}{3}+2\delta$, $\cdots$, $\frac{1}{3}+(s-1)\delta$, for some small $\delta > 0$, we can force $\mathcal{A}$ to open s new bins with a smaller maximum load at any time as follows. First release items of size $\frac{2}{3}-(s-1)\delta$ until $\mathcal{A}$ opens a new bin. At most $s + 1$ such items are required. We then let all items of size $\frac{2}{3}-(s-1)\delta$ depart except the last one packed in the new bin. Next we release items of size $\frac{2}{3}-(s-2)\delta$. At most s items are required to force $\mathcal{A}$ to open a new bin. We can repeat this process, for $1 \leq i \leq s$, with no more than $s-i+2$ items of size $\frac{2}{3}-(s-i)\delta$ to force $\mathcal{A}$ to open a new bin. At any time, the total load of all items not departed is at most $s + \frac{2}{3}$, versus $s(1 + \frac{2}{3})$ in the former case.

Based on the above observations, two operations are designed, namely, Op-Inc and Op-Comp. Op-Inc forces $\mathcal{A}$ to open bins each with one item of increasing size. Op-Comp then releases items of complementary size, ensuring that an item released in Op-Inc can be packed with a corresponding item released in Op-Comp into the same bin by an optimal offline algorithm.

Op-Inc and Op-Comp. The aim of Op-Inc is to make $\mathcal{A}$ open at least s more bins, for some $s > 0$, such that each new bin contains one item with item size increasing over the s bins. The items to be released have size in the range $[x, x+\epsilon]$, for some small ϵ, such that $x + \epsilon < \frac{1}{2}$. The adversary releases items of size x, $x + \frac{\epsilon}{s}$, $x + \frac{2\epsilon}{s}$, $\cdots$. Let $z_i = x + \frac{i\epsilon}{s}$. In each step i, the adversary releases z_i-items until $\mathcal{A}$ opens a new bin. We then let z_i-items depart except exactly one item of size z_i, for $0 \leq i < s$, in the i-th new bin opened by $\mathcal{A}$. At the end, Op-Inc opened s bins, each with increasing load from x to $x + \frac{(s-1)\epsilon}{s}$.

Op-Comp is designed to work with Op-Inc and assumes that there are s existing bins each with load in the range $[x, x + \frac{(s-1)\epsilon}{s}]$ where $x < x + \frac{(s-1)\epsilon}{s} < \frac{1}{2}$. The items to be released have size in the range $[1 - (x + \frac{(s-1)\epsilon}{s}), 1 - x]$. Starting from the largest load $x + \frac{(s-1)\epsilon}{s}$, in Step i, for $1 \leq i \leq s$, we release items of size $w_i = 1-(x + \frac{(s-i)\epsilon}{s})$ until $\mathcal{A}$ opens a new bin. Note that such items can only be packed in the first $s + 1 - i$ bins. We then let all w_i-items depart except the one packed in the new bin. Using these two operations, we can design an adversary to obtain the following lower bound.

Theorem 1 *No deterministic online algorithm can be better than* 8/3*-competitive.*

3 2-D and 3-D Online DBP of Unit Fraction Items

We extend the study of 2-D and 3-D online dynamic bin packing problem [13, 9] to unit fraction items. We adopt the typical approach of dividing items into classes and analyzing each class individually. The bin assignment algorithm that we use for 2-D and 3-D items is the First-Fit (FF) algorithm. When a new item R arrives, if there are occupied bins in which the item can be repacked, FF assigns R to the bin which has been occupied for the longest time.

For 2-D we consider the following classes of items. Class 1 contains $\mathrm{T}(\leq\frac{1}{3}, \leq 1)$-items. Class 2 contains types $\mathrm{T}(1, \leq\frac{1}{2}), \mathrm{T}(\frac{1}{2}, \leq\frac{1}{2})$. Class 3 contains types $\mathrm{T}(1, 1), \mathrm{T}(\frac{1}{2}, 1)$. Overall, the three classes cover all 2-D UF items and the overall competitive ratio (6.7850) is the sum of the competitive ratios of the three classes $(2.8258, 2.4593, 1.5$, respectively).

For 3-D we divide the items into Class 1 ($\mathrm{T}(> \frac{1}{2}, *, *)$-items), Class 2 ($\mathrm{T}(\leq \frac{1}{2}, > \frac{1}{2}, *)$-items), Class 3 ($\mathrm{T}(\leq \frac{1}{2}, (\frac{1}{3}, \frac{1}{2}], *)$-items), and Class 4 ($\mathrm{T}(\leq \frac{1}{2}, \leq \frac{1}{3}, *)$-items). The competitive ratios for the four classes are $6.7850, 4.8258, 4$, and 6, respectively.

Theorem 2 *There is a* 6.7850*-competitive algorithm and a* 21.6108*-competitive algorithm for* 2*-D and* 3*-D unit fraction items, respectively.*

References

[1] A. Bar-Noy and R. E. Ladner. Windows scheduling problems for broadcast systems. *SIAM J. Comput.*, 32:1091–1113, April 2003.

[2] A. Bar-Noy, R. E. Ladner, and T. Tamir. Windows scheduling as a restricted version of bin packing. *ACM Trans. Algorithms*, 3, August 2007.

[3] J. W.-T. Chan, T. W. Lam, and P. W. H. Wong. Dynamic bin packing of unit fractions items. *Theoretical Computer Science*, 409(3):172–206, 2008.

[4] J. W.-T. Chan, P. W. H. Wong, and F. C. C. Yung. On dynamic bin packing: An improved lower bound and resource augmentation analysis. *Algorithmica*, 53(2):172–206, 2009.

[5] E. G. Coffman, Jr., G. Galambos, S. Martello, and D. Vigo. Bin packing approximation algorithms: Combinatorial analysis. In *Handbook of Combinatorial Optimization*. Kluwer Academic Publishers, 1998.

[6] E. G. Coffman Jr., M. R. Garey, and D. S. Johnson. Dynamic bin packing. *SIAM J. Comput.*, 12(2):227–258, 1983.

[7] E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson. Bin packing approximation algorithms: A survey. In *Approximation Algorithms for NP-Hard Problems*, pages 46–93. PWS, 1996.

[8] J. Csirik and G. J. Woeginger. On-line packing and covering problems. In *On-line Algorithms—The State of the Art*, pages 147–177, 1996.

[9] L. Epstein and M. Levy. Dynamic multi-dimensional bin packing. *Journal of Discrete Algorithms*, 8:356–372, 2010.

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

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

[12] A. van Vliet. An improved lower bound for on-line bin packing algorithms. *Information Processing Letters*, 43(5):277–284, 1992.

[13] P. W. H. Wong and F. C. C. Yung. Competitive multi-dimensional dynamic bin packing via L-shape bin packing. In *WAOA 2009*, pages 242–254, 2010.