A Fundamental Restriction on Fully Dynamic Maintenance of Bin Packing[1]

Zoran Ivković[2] Errol L. Lloyd[3]

Abstract

This paper studies a fundamental restriction on the problem of maintaining an approximate solution for *one–dimensional bin packing* when items may arrive and depart dynamically. It is shown that imposing a fixed constant upper bound on the number of items that can be moved between bins per *Insert/Delete* operation forces the competitive ratio to be at least 4/3, regardless of the running time allowed per *Insert/Delete*. Thus, the ability to move more than a constant number of items is necessary for accomplishing highly competitive, time-efficient fully dynamic approximation algorithms for bin packing.

1 Introduction

In the (one–dimensional) bin packing problem, a list $L = (a_1, a_2, ..., a_n)$ of items where each item a_i has size $size(a_i) \in (0, 1]$ is given. The goal is to find the minimum k such that all the items can be packed into k unit–size bins. The reader is referred to [4] for background information and a survey of bin packing, together with a number of applications.

Recently, attention has been devoted to *on–line* and *dynamic* versions of bin packing [3, 4, 7, 10, 12]. These notions were extended to their full generality in [8] by considering *fully dynamic* bin packing, where:

- items may arrive and depart from the packing dynamically, and
- items may be moved from bin to bin as the packing is adjusted to accommodate arriving and departing items.

The algorithms presented in [8] process a sequence of *Inserts* (arrivals) and *Deletes* (departures) of items, as well as certain "lookup" queries that can be interspersed in the sequence of *Inserts* and *Deletes*.

Recall that the usual measure of the quality of a solution produced by a bin packing algorithm A is its *(relative) competitive ratio* $R(A)$ defined as:

$$R(A) = \lim_{n \to \infty} \sup_{OPT(L)=n} \frac{A(L)}{OPT(L)},$$

[1]Partially supported by the National Science Foundation under Grant CCR-9120731

[2]Yale School of Management, Yale University, New Haven, CT, 06520-8200, email: ivkovich@isis.som.yale.edu.

[3]Department of Computer and Information Sciences, University of Delaware, Newark, DE, 19716, email: elloyd@cis.udel.edu.

where $A(L)$ and *OPT(L)* denote, respectively, the number of bins used for packing list L by A and some optimal packing of L. We also say that A is $R(A)$–competitive.

In this paper, we prove a fundamental property of the fully dynamic bin packing problem. This property has important implications for the design of fully dynamic approximation algorithms for bin packing.

2 The Main Result

In this section we prove:

Theorem 1 *For any positive integer c, if A is a fully dynamic algorithm for bin packing that moves no more than c items (worst–case or amortized) per Insert/Delete operation, then the competitive ratio of A is at least* 4/3.

Proof: We establish that there are arbitrarily large lists L and suitably chosen sequences of *Inserts* and *Deletes* of elements of L, for which any A meeting the above conditions produces packings that utilize at least 4/3 of the optimal number of bins.

We define *B–items* as items with size greater than 1/2 and *B–bins* as bins containing a B–item.

Informally, a list that "defeats" A is constructed as follows:

1. Pick an arbitrary (large) integer M,
2. Take M B–items of size $1/2 + \epsilon$; construct M collections of items, where each collection contains a huge number (this number is a function of M, ϵ, and of course c) of exceedingly small items of size a, whose cumulative size is precisely $1/2 - \epsilon$,
3. Let L be a list of the items discussed in step (2); let A pack L in any order,
4. Observe the packing produced by A. If the excess bins amount to 1/3 of the optimal, A has failed. If not, we can then request deletions of all the B–items (of size $1/2+\epsilon$) from L and, consequently, from the packing. This sequence of deletions is then guaranteed to force A to utilize at least 4/3 the optimal number of bins for the resulting set of items.

The construction of L is aimed at defeating any algorithm A on the basis of A's inability to move more than c items across bins. Intuitively, A will fail not because it is too slow but because it is restricted in its movements, e.g., A is not unlike on–line algorithms in that there is a fixed bound on the number of items that can be moved across bins within each *Insert/Delete* operation.

We proceed with a formal proof of the theorem. First, pick an arbitrarily large positive integer M divisible by 6. Then, pick an arbitrarily small positive number ϵ such that $\epsilon < 1/M$. Finally, pick a (small) positive number a subject to the following three restrictions:

1. $\frac{1/2-\epsilon}{a}$ is an integer

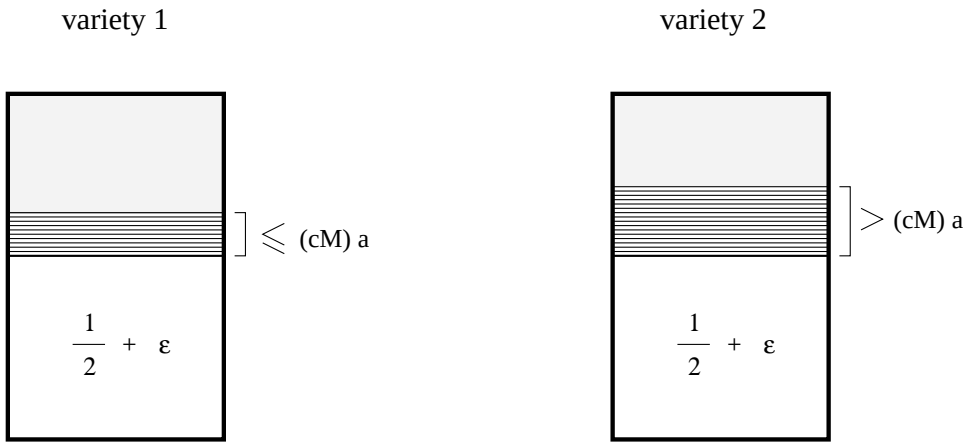

Figure 1: Two varieties of B–bins in the packing of L produced by A.

2. $a < \frac{\frac{3}{2M} - \epsilon}{cM}$

3. $a < \frac{\epsilon}{cM}$

The following list L defeats any fully dynamic algorithm A for bin packing that moves no more than c items (worst–case or amortized) per *Insert/Delete* operation:

$$L = (\underbrace{1/2+\epsilon, \ldots, 1/2+\epsilon}_{M}, \underbrace{a, \ldots, a}_{M\frac{1/2-\epsilon}{a}}).$$

Clearly, an optimal packing of L requires precisely M bins, where each bin contains a B–item of size $1/2 + \epsilon$ and $\frac{1/2-\epsilon}{a}$ items of size a.[4]

We fix A, let A pack L in any order, and then observe the packing of L produced by A. In this analysis, we distinguish between two varieties of B–bins in the packing produced by A: bins that, in addition to a B–item, contain at most cM items of size a – variety 1 B–bins, and bins that, in addition to a B–item, contain more than cM items of size a – variety 2 B–bins (see Figure 1).

The threshold value of cM was chosen so as to guarantee that any sequence of M changes to the packing via *Inserts/Deletes* of items would not result in the removal of all of the items of size a from any of the bins of variety 2, both in the case that the accounting of the number of items that A moves per operation is uniform and in the amortized case.

Note that the minimum gap size g (i.e., the unused portion of the bin) of a variety 1 B–bin is:

$$g = 1 - (1/2 + \epsilon + cMa) > 1/2 - \epsilon - cM\frac{\frac{3}{2M} - \epsilon}{cM} = \frac{1 - 3/M}{2}.$$

[4]Since we chose a so that $\frac{1/2-\epsilon}{a}$ is an integer, the level of each bin in an optimal packing is precisely 1, i.e., all of the M bins in an optimal packing are full.

Let the number of variety 1 B–bins in the packing produced by A be denoted by α, the number of variety 2 B–bins be denoted by β (note that $\alpha + \beta = M$), and the number of non–B–bins be denoted by γ.

Case 1: $\alpha \geq \frac{2}{3}M$. Due to the choice of the value of a ($a < \frac{\frac{3}{2M}-\epsilon}{cM}$), the cumulative size of all the gaps in all the variety 1 B–bins is at least:

$$\alpha g > \frac{2}{3}M\frac{1-3/M}{2} = M/3 - 1.$$

Thus, even under the assumption that all the variety 2 B–bins in the packing are full, there are still at least $M/3$ non–B–bins in the packing, i.e., $\gamma \geq M/3$.

Case 2: $\alpha < \frac{2}{3}M$. We know that $\beta > M/3$. Note that the cumulative size of all the non–B–items packed into a bin of variety 2 is at most $1/2 - \epsilon$.

To defeat A, we now request a sequence of M *Delete* operations that will delete all the B–items from L and the packing. Recall that none of the bins that were variety 2 B–bins before the deletions can be deleted from the packing, since each such bin contained more than cM items of size a. In addition, by our choice of a ($a < \frac{\epsilon}{cM}$), all these bins must have a level strictly less than $1/2$ at the conclusion of the deletion of all the B–items.

Note that the cumulative size of all the items in the list is now $M/2 - M\epsilon$ and the size of the optimal packing is now precisely $M/2$ (in order to guarantee the latter, we insisted that $\epsilon < 1/M$, i.e., $\epsilon M < 1$). Furthermore, the cumulative size of all the items in the β bins that were variety 2 B–bins (before the deletions) does not exceed $\beta/2$. Thus, A requires a certain number δ of additional bins to pack the entire (current) L:

$$\delta \geq M/2 - M\epsilon - \beta/2 > M/2 - 1/2 \cdot M/3 - M\epsilon > M/3 - 1.$$

A will require at least $\frac{2}{3}M$ bins to pack the (current) list L, since $\beta + \delta > \frac{2}{3}M - 1$. Since an optimal packing of (the current) L requires precisely $M/2$ bins, A was forced to utilize at least $4/3$ of the number of bins utilized by an optimal packing.

□

3 Discussion

In light of Theorem 1, we address the need to move items between bins in response to *Insert*s and *Delete*s. The difficulties related to this issue are twofold. First, such moves have to be carried out in a manner that would guarantee small competitive ratios. Second, all the moves have to be carried out within low running times (i.e., $o(n)$ per *Insert* or *Delete*, since off–line algorithms that achieve any competitive ratio greater than 1 in $\mathcal{O}(n)$ running time are known[6]).

Intuitively, difficulties may arise while handling very small items: the attempt to move a large number of very small items from a bin, item by item, could result in a prohibitively large running time.

A natural question is whether there are fully dynamic algorithms for bin packing that are allowed to move $\omega(1)$ items[5] per *Insert/Delete* operation with a competitive ratio of less than 4/3, i.e., would removing the restriction on the number of items that may be moved per operation help? The results obtained in [8] show that the answer to this question is yes.

The main result reported in [8] is a 5/4–competitive fully dynamic algorithm *MMP* (Mostly Myopic Packing). *MMP* processes *Inserts* and *Deletes* of items in $\Theta(\log n)$ worst-case running time. In *MMP*, the efficient manipulation of very small items is accomplished via **bundling**. The purpose of bundles is to allow the efficient manipulation of large numbers of very small items at one time: rather than moving these very small items from a bin/to a bin individually, the algorithm moves entire bundles of very small items.[6] Moving an entire bundle can be accomplished within the same running time as moving a single larger item.

Thus, allowing/disallowing the moving of $\omega(1)$ items between bins per *Insert/Delete* operation has a crucial impact on the competitive ratio of fully dynamic approximation algorithms for bin packing.

Finally, there are two major open questions. First, is there an algorithm for a restricted version of fully dynamic bin packing (where the number of items that can be moved between bins per *Insert/Delete* operation is bounded by a constant) with a constant competitive ratio, and, if so, is that competitive ratio close to 4/3? Second, is there a better lower bound than 4/3? In the case of on-line bin packing, dealing only with *Insert*s, somewhat stronger lower bounds are known: Yao proved a 3/2 bound [12], and Brown [2] and Liang [9] improved that to 1.536.... Similar results may be possible for the fully dynamic case.

[5] ω-notation is defined as follows: $f(n) \in \omega(g(n))$ if and only if $g(n) \in o(f(n))$ [5].

[6] Note that this idea was used, albeit in different contexts, in [1] and [7].

References

[1] R. J. Anderson, E. W. Mayr, and M. K. Warmuth. (1989). Parallel approximation algorithms for bin packing. *Information and Computation* **82**, 262–277.

[2] D. J. Brown. (1979). A lower bound for on-line one-dimensional bin packing algorithms. Technical Report R-864, Coordinated Science Laboratory, University of Illinois, Urbana, IL.

[3] E. G. Coffman, M. R. Garey, M. R., and D. S. Johnson. (1983). Dynamic bin packing. *SIAM Journal on Computing* **12**, 227–258.

[4] E. G. Coffman, M. R. Garey, and D. S. Johnson. (1984). Approximation algorithms for bin packing: an updated survey. In *Algorithm Design for Computer System Design* (G. Ausiello, M. Lucertini, and P. Serafini, Eds.), 49–106. Springer–Verlag, New York.

[5] T. H. Corman, C. E. Leiserson, and R. L. Rivest. (1990). *Introduction to Algorithms.* The MIT Press, Cambridge, MA.

[6] W. Fernandez de la Vega and G. S. Lueker. (1981). Bin packing can be solved within $1 + \epsilon$ in linear time. *Combinatorica* **1**(4), 349–355.

[7] G. Gambosi, A. Postiglione, and M. Talamo M. (1990). New algorithms for on–line bin packing. In *Algorithms and Complexity, Proceedings of the First Italian Conference*, (G. Aussiello, D. P. Bovet, and R. Petreschi, Eds.), 44–59. World Scientific, Singapore.

[8] Z. Ivković and E. L. Lloyd. (1993). Fully Dynamic Algorithms for Bin Packing: Being (Mostly) Myopic Helps. *Proceedings of the 1st European Symposium on Algorithms*, 224–235. Lecture Notes in Computer Science No. 726, Springer–Verlag, New York. To appear in the *SIAM Journal of Computing.*

[9] F. M. Liang (1980). A lower bound for on–line bin-packing. *Information Processing Letters* **10**, 76–79.

[10] C. C. Lee and D. T. Lee. (1985). A simple on–line bin–packing algorithm. *Journal of the ACM* **3**, 562–572.

[11] P. Ramanan, D. J. Brown, C. C. Lee, and D. T. Lee. (1989). On-line bin-packing in linear time. *Journal of Algorithms* **3**, 305–326.

[12] A. C.–C. Yao. (1980). New algorithms for bin packing. *Journal of the ACM* **27**(2), 207–277.