An approximation algorithm for square packing

Rob van Stee*

February 18, 2004

Abstract

We consider the problem of packing squares into bins which are unit squares, where the goal is to minimize the number of bins used. We present an algorithm for this problem with an absolute worst-case ratio of 2, which is optimal provided P $\neq$ NP.
Keywords: bin packing, square packing, approximation algorithm, absolute worst-case ratio

1 Introduction

Bin packing is one of the oldest and most well-studied problems in computer science [6, 4]. In this paper, we study a natural generalization of bin packing, called square packing. In this problem, we receive a sequence σ of *squares* $p_1, p_2, \ldots, p_n$. We define the *size* of a square p, denoted by $s(p)$, as the length of one of its edges. We have an infinite number of *bins*, each of which is a unit square. Each item must be assigned to a bin and a position $(x_1(p), x_2(p))$, where $0 \leq x_i(p)$ and $x_i(p) + s(p) \leq 1$ for $i = 1, 2$. Further, the positions must be assigned in such a way that no two items in the same bin overlap. A bin is *empty* if no item is assigned to it, otherwise it is *used*. The goal is to minimize the number of bins used.

Most of the previous work on bin packing has focused on the *asymptotic performance ratio* (approximation ratio), which we now define. For a given input sequence σ, let $\text{cost}_{\mathcal{A}}(\sigma)$ be the number of bins used by algorithm $\mathcal{A}$ on σ. Let $\text{cost}(\sigma)$ be the minimum possible number of bins used to pack items in σ. The *asymptotic performance ratio* for an algorithm $\mathcal{A}$ is defined to be

$$\mathcal{R}_{\mathcal{A}}^{\infty} = \limsup_{n\to\infty} \sup_{\sigma} \left\{ \frac{\text{cost}_{\mathcal{A}}(\sigma)}{\text{cost}(\sigma)} \middle| \text{cost}(\sigma) = n \right\}.$$

Thus the focus here is on the long-term behavior of algorithms. In contrast, in the current paper we consider the *absolute* worst-case ratio [13, 15], which for an algorithm $\mathcal{A}$ is defined as follows:

$$\mathcal{R}_{\mathcal{A}} = \sup_{\sigma} \frac{\text{cost}_{\mathcal{A}}(\sigma)}{\text{cost}(\sigma)}.$$

Attaining a good absolute worst-case ratio is more difficult than attaining a good asymptotic worst-case ratio, because in the second case an algorithm is allowed to "waste" a constant number of bins, which allows e.g. the classification of items followed by a packing where each class is packed separately.

*Centre for Mathematics and Computer Science (CWI), Kruislaan 413, NL-1098 SJ Amsterdam, The Netherlands. Rob.van.Stee@cwi.nl. Work supported by the Netherlands Organization for Scientific Research (NWO), project number SION 612-061-000.

Previous Results: For a survey on bin packing, see [4]. The most prominent results are as follows: Garey, Graham and Ullman [9] were the first to study the approximation ratios of both online and offline algorithms. Fernandez de La Vega and Lueker [7] presented the first approximation scheme for bin packing. Karmarkar and Karp [10] gave an algorithm which uses at most $\text{cost}(\sigma) + \log^2(\text{cost}(\sigma))$ bins.

The problem of rectangle packing, which is a generalization of square packing where the items to be packed are rectangles, was first introduced by Chung, Garey and Johnson [3]. Caprara [2] presented an algorithm with approximation ratio Π_∞. For square packing, Ferreira, Miyazawa and Wakabayashi give a 1.988-approximation algorithm [8]. Approximation schemes were presented by Bansal and Sviridenko [1] and Correa and Kenyon [5] independently.

A variation on this problem was considered by Schiermeyer [13] and Steinberg [15]. They studied the problem of packing rectangles into a strip with unit width and unbounded height so as to minimize the total height of the packing, and independently gave algorithms with an absolute worst-case ratio of 2.

Leung et al. [11] showed that it is NP-hard to determine whether or not a given set of squares can be packed in a single bin. This implies that there cannot be a polynomial-time algorithm with an absolute worst-case ratio less than 2, unless P = NP. Such an algorithm could be used to determine (in polynomial time) whether a set of squares fits into a single bin: for a given set of items, if that algorithm packs them into two bins, they cannot be packed in a single bin because the absolute worst-case ratio is strictly less than 2.

Our Result: We present an algorithm for square packing with an absolute worst-case ratio of 2, which is optimal provided P $\neq$ NP.

2 Subroutines for the algorithm

We classify items according to their size. *Huge* items have size greater than $2/3$. *Big* items have size in $(1/2, 2/3]$. *Medium-sized* items have size in $(1/3, 1/2]$. Finally, *small* items have size at most $1/3$.

All the non-small items will be packed using the algorithm FIRST FIT DECREASING SIZE (FFDS). This algorithm works as follows. First, huge and big items are packed into bins: one item per bin, in order of increasing size. Each item is placed in a corner of its bin. This gives a list B of bins. Next, the medium-sized items are sorted in order of decreasing size, giving a list L'.

The algorithm now does the following repeatedly. It checks whether the first three items of L' can be packed together with the first bin in B, i.e. the one that contains the smallest big item. If the three items fit there, they are placed there; otherwise the first four items from L' are put in a new, empty bin. The packed items are then removed from L', the first bin is removed from B if it was used to pack them, and the algorithm continues in the same way. If $B = \emptyset$ at some point, the remaining items in L' are packed four to a bin in new bins, until all items are packed.

Ferreira, Miyazawa, and Wakabayashi [8] define FFDS and prove the following.

Lemma 1 (Ferreira, Miyazawa, Wakabayashi [8]) *Let L be a list of squares that all have size greater than $1/3$. Then the algorithm FFDS applied to L generates a packing where each bin, except possibly one, contains*

- *One big or huge item and no medium-sized items, or*
- *One big item and three medium-sized items, or*
- *Four medium-sized items*

The remaining bin, if there is one, contains at most three items, including at most one big item. The packing that FFDS generates is an optimal packing for L.

To pack the small items, we will use the algorithm NEXT FIT DECREASING (NFD) as a subroutine. The version of the algorithm considered here packs squares into a rectangle of size $a \times b$. The idea of this algorithm is very simple. First we sort the squares into non-increasing order. We pack items into slices. The width of a slice is b. We use NEXT FIT on the sorted list of items, considering the slices as bins. When a new slice is allocated, its height is set equal to the height of the first item placed in it. Since the items are packed in order of non-increasing size, subsequent items fit in the slice. Slices are allocated from the rectangle going from bottom to top. The algorithm halts either when all items are packed, or when it is impossible to allocate a slice. In the later case, some items remain unpacked.

Meir and Moser [12] introduce NEXT FIT DECREASING and prove the following Lemma:

Lemma 2 (Meir & Moser [12]) *Let L be a list of squares with sides $x_1 \geq x_2 \geq \ldots$ Then L can be packed in a rectangle of height $a \geq x_1$ and width $b \geq x_1$ using* NEXT FIT DECREASING *if one of the following conditions is satisfied:*

- *the total area of items in L is at most $x_1^2 + (a - x_1)(b - x_1)$.*
- *the total area of items in L is at most $ab/2$.*

3 Algorithm

In this section, we give a detailed description of the algorithm. We start by applying the algorithm FFDS from [8] to the items of size greater than $1/3$. After this, only the small items remain to be packed. These items are packed in three steps. If at some point during these three steps, all small items are already packed, the algorithm halts.

1. Bins containing only a big item and no medium-sized items are filled further with small items.
2. A bin containing at most three medium-sized items, or a big item and at most two medium-sized items, is used if it exists. There can be at most one such bin by Lemma 1.
3. Finally, if there are still small items left, they are packed into bins by themselves.

The details of these three steps are described below.

Step 1 *Bins with one big item, but no medium-sized items.* The big item is placed into the lower left corner of the bin. Denote its size by x. The remaining area can be divided into two rectangles, one of dimensions 1 by $1 - x$ at the top of the bin and one of dimensions $1 - x$ by x next to the big item. Use Next Fit Decreasing to pack the first rectangle. Continue until an item can no longer be placed, place that item in the second rectangle. Note that this is possible since the big item has size at most $2/3$ and small items have size at most $1/3$.

Step 2A *A bin with only one medium-sized item.* Pack items as in Step 1, the medium-sized item in the lower left corner.

Step 2B *A bin with two items, at least one medium-sized.* Place the largest item, of size x_1, in the lower left corner of the bin. Place the second largest item next to it, aligned with the bottom of the bin and as far to the left as possible. There is an unoccupied region of dimensions 1 by $1 - x_1$ at the top of the bin. Pack small items into this region using NFD.

Step 2C *A bin with three items, at least two medium-sized.* Place the first two items as in Step 2B. Place the third item, of size x_3, on top of the first one, aligned with the left edge of the bin and as far down as possible. This leaves an unoccupied region of dimensions $1 - x_3$ by $1 - x_1$ in the top right corner of the bin. Pack small items into this region using NFD.

Step 3 *Bins with only small items.* Pack items into new bins using NFD, opening a new bin whenever items can no longer be placed in the current bin.

4 Worst-case ratio

Lemma 3 *If there are any unpacked small items left after Step 2, each bin that is packed so far contains a total area of at least* $4/9$.

Proof The lemma clearly holds for any bins with huge items.

Bins that are packed in Step 1 or 2 are packed exactly as in the proof of Lemma 4.3 in [14]. Step 1 and 2A correspond to Case 2 from that proof, Step 2B corresponds to Case 4 and 2C corresponds to Case 5.

It follows immediately from that proof that in all cases, the used area is at least $4/9$. □

Lemma 4 *Consider a bin that is packed in Step 3. If after packing this bin, there are still unpacked small items left, it contains a total area of at least* $9/16$.

Proof We distinguish between cases. Since all items have size at most $1/3$, at least nine items can be packed together in the bin, and NFD allocates at least three slices.

Case 1 *The first slice contains at least four items.*

Denote the size of the largest item by x, the size of the largest item in the second slice by y and the size of the first item that can no longer be placed by z. Denote the total area of items starting from the second slice by f. By Lemma 2, we have $f + z^2 > y^2 + (1-y)(1-x-y)$. Therefore in the entire bin we pack at least

$$x^2 + 4y^2 + (1-y)(1-x-y) - z^2 \geq x^2 + 3y^2 + (1-y)(1-x-y) = g.$$

We have $\frac{\partial g}{\partial x} = 2x + y - 1$, which is negative for $0 \leq x < 1/3$ and $0 \leq y \leq 1/3$. We find that g has a minimum of $29/48 > 9/16$ for $\{(x,y) \in \mathbb{R}^2 | 0 \leq x \leq 1/3, 0 \leq y \leq 1/3\}$, which is attained for $x = 1/3$ and $y = 5/24$.

In the remaining cases, the four largest items do not fit next to each other in a bin.

Case 2 *The first slice contains three items, but the second slice contains at least four items.*

Denote the sizes of the first items in the first three slices by x, y and z, respectively. Denote the sizes of the other items in the first slice by x_1, x_2. Again using Lemma 2, we pack at least

$$x^2 + x_1^2 + x_2^2 + y^2 + 3z^2 + (1-z)(1-x-y-z).$$

We are interested in its minimum under the conditions that $0 < z \leq y \leq x_2 \leq x_1 \leq x \leq 1/3$ and $x + x_1 + x_2 + y \geq 1$. By distinguishing between the cases $y \leq 2/9$ and $y > 2/9$, we find that this function has a minimum of $743/1296 > 0.5733 > 9/16$ which is attained for $x = 1/3$, $x_1 = x_2 = y = 2/9$ and $z = 13/72$.

Case 3 *The first two slices both contain three items.*

Denote the size of the largest item by x, the size of the largest item in the second slice by y and the size of the second largest item in the second slice by z. By Lemma 2, any set of squares with total area at most $(1-x-y)/2$ can be packed by NFD starting from the third slice. Thus NFD packs at least $(1-x-y)/2-z^2$ in that region, and in total at least

$$x^2 + 3y^2 + z^2 + (1 - x - y)/2.$$

We have $x > 1/4$, $y > 1/4$, and $z > (1 - y)/3$ since there are only three items in the first two slices. The expression is monotonically increasing in x, y and z on this domain, and we find that it is at least $9/16$ (attained for $x = y = z = 1/4$). □

Theorem 1 *The algorithm has an absolute worst-case ratio of 2.*

Proof Denote the number of bins with a huge item by h, the number of bins that have a big item (but no medium-sized items) by b, the number of bins with medium-sized items (and possibly a big item) by m and the number of bins with (only) small items of total area at least $9/16$ by s. Our algorithm may generate one bin (the last one) that has only small items but with total area less than $9/16$. Thus the number of bins produced by the algorithm is at most

$$h + b + m + s + 1.$$

Since FFDS is an optimal algorithm, for the optimal solution OPT we find $\text{OPT} \geq h + b + m$. Thus as long as $s + 1 \leq h + b + m$, our algorithm uses at most twice as many bins as an optimal solution.

Suppose $s \geq h + b + m$. First of all, if $h + b + m = 0$, then by Lemma 4 we have $\text{OPT} \geq \frac{9}{16}s$. If $s = 0$, then $\text{OPT} = 1$ (assuming nonzero input) and the algorithm is optimal. Otherwise, $\text{OPT} > s/2$, and therefore $\text{OPT} \geq (s + 1)/2$.

Suppose $s \geq h + b + m \geq 1$. This implies that there are bins packed with only small items. In other words, we do not run out of items while packing small items in Steps 1 or 2. Lemma 3 guarantees that in this case, all bins packed so far contain a total area of at least $4/9$. Furthermore, all bins with only small items, except possibly the last one, have total area at least $9/16$ by Lemma 4.

Thus in the case that $s \geq h + b + m$, each bin except possibly the last one is on average strictly more than half full, since $4/9 + 9/16 > 1$ and $s \geq 1$. (Note that if the last bin with only small items does not contain at least an area of $9/16$, it is not counted in s.) This implies that any packing of this input requires strictly more than $(h+b+m+s)/2$ bins, and therefore at least $(h+b+m+s+1)/2$ bins. This concludes the proof. □

References

[1] Nikhil Bansal and Maxim Sviridenko. New approximability and inapproximability results for 2-dimensional packing. In *Proceedings of the 15th Annual Symposium on Discrete Algorithms*, pages 189–196. ACM/SIAM, 2004.

[2] Alberto Caprara. Packing 2-dimensional bins in harmony. In *Proc. 43th IEEE Symp. on Found. of Comp. Science*, pages 490–499, 2002.

[3] Fan R. K. Chung, Michael R. Garey, and David S. Johnson. On packing two-dimensional bins. *SIAM Journal on Algebraic and Discrete Methods*, 3:66–76, 1982.

[4] Edward G. Coffman, Michael R. Garey, and David S. Johnson. Approximation algorithms for bin packing: a survey. In D. Hochbaum, editor, *Approximation algorithms*. PWS Publishing Company, 1997.

[5] Jose Correa and Claire Kenyon. Approximation schemes for multidimensional packing. In *Proceedings of the 15th ACM/SIAM Symposium on Discrete Algorithms*, pages 179–188. ACM/SIAM, 2004.

[6] János Csirik and Gerhard J. Woeginger. On-line packing and covering problems. In *A. Fiat and G. J. Woeginger, editors*, Online Algorithms: The State of the Art, volume 1442 of *Lecture Notes in Computer Science*, pages 147–177. Springer-Verlag, 1998.

[7] Wenceslas Fernandez de la Vega and George S. Lueker. Bin packing can be solved within $1 + \varepsilon$ in linear time. *Combinatorica*, 1:349–355, 1981.

[8] Carlos E. Ferreira, Flavio K. Miyazawa, and Yoshiko Wakabayashi. Packing squares into squares. *Pesquisa Operacional*, 19(2):223–237, 1999.

[9] Michael R. Garey, Ronald L. Graham, and Jeffrey D. Ullman. Worst-case analysis of memory allocation algorithms. In *Proceedings of the Fourth Annual ACM Symposium on Theory of Computing*, pages 143–150. ACM, 1972.

[10] Narendra Karmarkar and Richard M. Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, pages 312–320, 1982.

[11] J.Y.T. Leung, T.W. Lam, C.S. Wong, G.H. Young, and F.Y.L. Chin. Packing squares into a square. *Journal on Parallel and Distributed Computing*, 10:271–275, 1990.

[12] A. Meir and L. Moser. On packing of squares and cubes. *Journal of Combinatorial Theory*, 5:126–134, 1968.

[13] Ingo Schiermeyer. Reverse-fit: a 2-optimal algorithm for packing rectangles. In *Algorithms - ESA '94, Proceedings Second Annual European Symposium*, pages 290–299, 1994.

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

[15] A. Steinberg. A strip-packing algorithm with absolute performance bound 2. *SIAM Journal on Computing*, 26(2):401–409, April 1997.