Packing Squares into Squares*

Carlos E. Ferreira[†], Flávio K. Miyazawa[‡] and Yoshiko Wakabayashi[†]

August 1999

Abstract

We consider the problem of packing a list of squares into unit capacity squares. The objective of this problem is to minimize the number of unit capacity squares used in the packing. We consider a restricted version of this problem and exhibit a polynomial time algorithm to solve it. We use this algorithm in the design of an approximation algorithm for the original problem, and show that its asymptotic performance bound is 1.988. This bound compares favourably with the bound 2.125, known for an algorithm obtained by Chung, Garey and Johnson for the more general problem, where the items to be packed are rectangles.

1 Introduction

We consider the problem of packing squares into squares, defined as follows. Given a list L of n squares and (an unlimited quantity of) squares Q, called *bins*, pack the n squares of L into a minimum number of bins. We note that there is no loss of generality to assume that the bins have unit capacity (width and height 1), since otherwise we can scale the squares in L. So we assume henceforth that the bins have unit capacity (we call them unit bins) and denote this problem as SPP (*Square Packing Problem*).

The packings we consider are all orthogonal. That is, with respect to a fixed side of the bin, the sides of the squares must be parallel or orthogonal to it.

This problem has applications in stock-cutting, VLSI chip design and scheduling programs in parallel computers [17, 18]. Other problems related to SPP that have been mentioned in the literature are the following. Erdős and Graham [11] studied the problem of finding the maximum number of equal squares that can be packed in a rectangle; and Göbel [13] investigated the problem of finding the smallest possible square into which n unit squares can be packed (see also [12]). These two problems deal with non-necessarily orhtogonal packings. For orthogonal packings, Meir and Moser presented in [19] an algorithm which—under certain conditions—finds an optimum packing of a list of squares into a given rectangle, provided that the area of the rectangle satisfies certain size condition. Coffman and Lagarias [5, 6] considered probabilistic analyses of algorithms for the problem of packing squares in a strip with width 1 and unlimited

*This work has been partially supported by Project ProNEx 107/97 (MCT/FINEP), Project PCE (ProTeM-CC-III/CNPq), FAPESP (Proc. 96/04505-2) and CNPq individual research grants: Proc. 300752/94-6 (cef@ime.usp.br), Proc. 300301/98-7 (fkm@dcc.unicamp.br), Proc. 304527/89-0 (yw@ime.usp.br).

[†]Instituto de Matemática e Estatística — Universidade de São Paulo — Rua do Matão, 1010 – Cidade Universitária — 05508–900 — São Paulo–SP — Brazil.

[‡]Instituto de Computação — Universidade Estadual de Campinas — C. Postal 6176 — 13083-970 — Campinas–SP — Brazil.

height. In this problem the objective is to minimize the size of the packing in the unlimited direction. In [1], Baker *et al.* presented approximation algorithms for the problem of maximizing the number of squares (of a given set) that can be packed into a rectangle.

For other related problems, applications and an updated literature on cutting and packing, the reader is referred to [3, 9, 10].

It is known that SPP is an $\mathcal{NP}$-hard problem. This is a consequence of a result of a special case of this problem that is already $\mathcal{NP}$-complete. Leung *et al.* [17] have shown that the problem of deciding whether a given set of squares can be packed into only one square is $\mathcal{NP}$-complete in the strong sense.

One approach to deal with $\mathcal{NP}$-hard problems is to develop approximation algorithms. These are polynomial time algorithms that have some (worst-case) performance guarantee compared to the optimum solution. In the case of packing problems, there are two standard metric for worst-case performance: the absolute performance bound and the asymptotic performance bound. These concepts are defined as follows.

For a given list L of items to be packed, and algorithm $\mathcal{A}$, let $\mathcal{A}(L)$ be the number of bins used when algorithm $\mathcal{A}$ is applied to the list L; and let $\mathrm{OPT}(L)$ denote the optimum number of bins for a packing of L.

We say that an algorithm $\mathcal{A}$ has *absolute performance bound* α if

$$\mathcal{A}(L) \leq \alpha \cdot \mathrm{OPT}(L), \qquad \text{for all input list } L;$$

and we say that $\mathcal{A}$ has *asymptotic performance bound* α if there exists a constant β such that

$$\mathcal{A}(L) \leq \alpha \cdot \mathrm{OPT}(L) + \beta, \qquad \text{for all input list } L.$$

In 1982, Chung, Garey and Johnson [2] presented an algorithm for the problem of packing a list L of rectangles into a minimum number of unit (square) bins. This algorithm, called HFF (Hybrid First Fit), has asymptotic performance bound 2.125. More precisely, these authors proved the following inequality:

$$\mathrm{HFF}(L) \leq 2.125 \cdot \mathrm{OPT}(L) + 5, \qquad \text{for all input list } L.$$

Since SPP is a special case of the above problem, the algorithm HFF can also be used to solve SPP. Thus we may view HFF as an algorithm for SPP with asymptotic performance bound 2.125.

We should mention that for the problem of packing a list of squares in a strip of fixed width and unlimited height, with the objective of minimizing the packing in the unlimited direction, Coffman, Garey, Johnson and Tarjan [4] have shown that the algorithm FFDH (First Fit Decreasing Height) has a tight asymptotic performance bound 1.5.

In the next section we present a proof that there cannot exist an approximation algorithm for SPP with absolute performance bound less than 2, unless $\mathcal{P} = \mathcal{NP}$. In Section 3 we study a special case of SPP that can be solved in polynomial time. In Section 4 we present an approximation algorithm for SPP with asymptotic performance bound 1.988. Finally, Section 5 brings some concluding remarks.

2 Non-approximability in the absolute sense

In this section we present a result concerning the impossibility of obtaining a polynomial time approximation algorithm for SPP with certain absolute performance bound (under the assumption that $\mathcal{P} \neq \mathcal{NP}$). First, we introduce a special decision version of SPP, denoted by PSU^d, defined as follows: given a list L of squares, decide whether all squares of L can be packed into a unit bin. Leung *et al.* [17] have shown the following result with respect to the complexity of PSU^{d}.

Theorem 2.1. PSU^d *is strongly* $\mathcal{NP}$*-complete.*

Using this result we can prove that finding a polynomial time algorithm for SPP with absolute performance bound less than 2 is as difficult as the original problem.

Theorem 2.2. *There is no polynomial time algorithm for* SPP *with absolute performance bound less than 2, unless* $\mathcal{P} = \mathcal{NP}$.

Proof. Suppose that there exists a polynomial time algorithm $\mathcal{A}$ for SPP such that

$$\mathcal{A}(L) < 2 \cdot \text{OPT}(L) \quad \text{for all input list } L \text{ } of \text{ SPP}.$$

Consider now an instance L of PSU^d and apply algorithm $\mathcal{A}$ to L. We have two possibilities. If $\mathcal{A}(L) < 2$, then we have a positive answer for PSU^d. If $\mathcal{A}(L) \geq 2$, then the answer for PSU^d is negative. Thus the algorithm $\mathcal{A}$ solves PSU^d in polynomial time, contradicting Theorem 2.1. □

3 Packing squares with restricted sizes

We call *size* of a square $q \in L$, denoted by $s(q)$, the size of its width (or height). Here it is assumed that every square in the list L has size at most 1.

Since it is not possible (unless $\mathcal{P} = \mathcal{NP}$) to find approximation algorithms for SPP with absolute performance bound less than 2, one could try to develop polynomial time algorithms for restricted instances of the problem. The question that arises is: what restrictions can we impose on the instances such that the restricted problem becomes polynomially solvable?

One possibility is to restrict the sizes of the squares in L to some values, as follows. For each integer $k \geq 2$, denote by $\text{SPP}(k)$ the restricted version of SPP whose instance is a list L consisting of squares of size greater than $\frac{1}{k}$; that is, $L = (q_1, q_2, \ldots, q_n)$, where $s(q_i) > \frac{1}{k}$.

It is immediate that SPP(2) can be easily solved. Let us consider SPP(3). In the sequel we describe an algorithm for SPP(3), called FFDS (First Fit Decreasing Size). After that, we prove a result on the configurations of the bins generated by this algorithm (needed later in the other algorithm), and then we prove that FFDS solves SPP(3).

Algorithm FFDS

Input: List $L = (q_1, \ldots, q_n)$ of squares, where $\frac{1}{3} < s(q_i) \leq 1$.
Output: Packing of L into a minimum number of unit bins.

1. Let $L_1 \leftarrow \{q \in L : \ s(q) > \frac{1}{2}\ \}$ and $L_2 \leftarrow L \setminus L_1$.

2. Sort the squares of L_1 in increasing order of their sizes.
Let $L_1 = \{q_1, \ldots, q_p\}$, with $s(q_1) \leq \ldots \leq s(q_p)$.

3. For $i \leftarrow 1, \ldots, p$ take a unit bin, say Q_i, and pack q_i into it left-justified at the bottom corner (see Figure 1, configuration $C2$).

4. Sort the squares of L_2 in decreasing order of their sizes.
Let $L_2 = \{q_{p+1}, \ldots, q_n\}$, with $s(q_{p+1}) \geq \ldots \geq s(q_n)$.

5. Let $i \leftarrow p + 1$; $j \leftarrow 1$ and $r \leftarrow 0$.

6. While $i \leq n$

If $j \leq p$ then
check whether q_i can be packed in the (partially occupied) bin Q_j.

6.1 If yes, pack q_i, q_{i+1}, q_{i+2} in Q_j adjusting them to each of the non-occupied corners (both or one of the squares q_{i+1}, q_{i+2} may not exist; pack only the existing ones); let $i \leftarrow i + 3$; $j \leftarrow j + 1$.

6.2 If not, let $r \leftarrow r + 1$ and pack $q_i, q_{i+1}, q_{i+2}, q_{i+3}$ (the existing ones) in a new bin Q_{p+r}; let $i \leftarrow i + 4$.

else (there is no partially occupied bin available)
let $r \leftarrow r + 1$ and pack $q_i, q_{i+1}, q_{i+2}, q_{i+3}$ (only the existing ones) in a new bin Q_{p+r}; let $i \leftarrow i + 4$.

7. Return the packing (using $p + r$ bins) that was generated. Halt.

end algorithm

Lemma 3.1. *Let L be an instance of* SPP(3), $L_1 := \{q \in L : \ s(q) > \frac{1}{2}\ \}$ *and* $L_2 := L \setminus L_1$. *Then the algorithm* FFDS *applied to L generates a packing where each bin, except possibly one, has one of the following configurations.*

$C1$: *configuration consisting of 1 square of L_1 and 3 squares of L_2.*

$C2$: *configuration consisting of exactly 1 square of L_1.*

$C3$: *configuration consisting of 4 squares of L_2.*

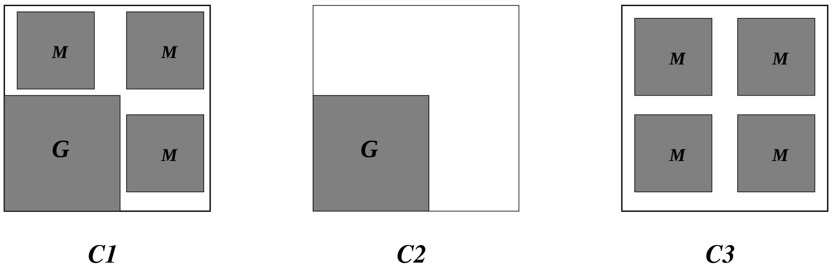

Figure 1: Main configurations of the bins generated by the algorithm FFDS.

Proof. First, we note that the algorithm FFDS generates a packing where the bins may have configuration $C1$ (see step 6.1 of the algorithm), or $C2$ (step 3) or $C3$ (step 6.2 and step 6 when

$j > p$). A bin having a configuration different from $C1$, $C2$ or $C3$ may occur only if in step 6 the variable i is at most n and it becomes for the first time larger than $n-3$, and one of the two possible situations occurs.

- The square q_i can be packed in a partially occupied bin Q_j (step 6.1), but not both of the squares q_{i+1}, q_{i+2} exist (this means that $i \geq n-1$). In this case, the bin Q_j changes its configuration from $C2$ to a configuration consisting of 1 square of L_1 and 1 or 2 squares of L_2. This happens only once (since afterwards $i > n$).
- The square q_i cannot be packed in a partially occupied bin Q_j (step 6.2 or when $j > p$). In this case, since $i > n-3$ this means that not all of the squares $q_{i+1}, q_{i+2}, q_{i+3}$ exist. Thus, a bin having a configuration consisting of precisely 1, 2 or 3 squares of L_2 is generated. Here again, afterwards we have $i > n$ and the algorithm halts.

If at the mentioned stage none of the two situations above occurs (this may happen when step 6.1 is performed with $i = n-2$ and both q_{i+1} and q_{i+2} exist), then no bin will have a configuration different from $C1$, $C2$ or $C3$. Thus, the algorithm FFDS generates a packing as claimed in the lemma. □

Theorem 3.2. *The algorithm* FFDS solves SPP(3) in polynomial time.

Proof. Let L be an instance of SPP(3) consisting of squares with size greater than $\frac{1}{3}$. Let $L_1 := \{q : s(q) > \frac{1}{2}\}$ and $L_2 := L \setminus L_1$. The proof is by induction on $|L|$. If $L_1 = \emptyset$ or $|L_2| \leq 4$ the result follows immediately. Let us assume that $L_1 = \{q_1, \ldots, q_p\}$, $p \geq 1$, and $L_2 = \{q_{p+1}, \ldots, q_n\}$, $|L_2| > 4$, with the squares ordered as in steps 2 and 3 of the algorithm FFDS.

First we note that in any packing of L each of the squares in L_1 must be packed in a different bin. Without loss of generality call these bins $Q_1, \ldots, Q_p$ as in the algorithm (with q_i packed in Q_i). We consider two cases.

Case 1. The square q_{p+1} cannot be packed in Q_1.

Case 2. The square q_{p+1} can be packed in Q_1.

In case 1, let $R := \{q_{p+1}, \ldots, q_{p+4}\}$ and $L' := L \setminus R$. In case 2, let $R := \{q_{p+1}, q_{p+2}, q_{p+3}\}$ and $L' := L \setminus (\{q_1\} \cup R)$.

In case 1 the algorithm FFDS applied to L generates a packing that consists of the solution of FFDS applied to L' together with an additional bin having configuration $C3$ (containing the 4 squares in R). In case 2 the algorithm FFDS applied to L generates a packing that consists of the solution of FFDS applied to L' together with an additional bin having configuration $C1$ (containing q_1 and the 3 squares in R).

In both cases, by induction hypothesis, the algorithm FFDS applied to L' generates an optimum packing of L'. Let us assume that this optimum packing uses $t-1$ bins. Thus, the packing generated by FFDS applied to L uses t bins.

It remains to show that any optimum packing of L uses t bins. For that, suppose there is an optimum packing of L that uses less than t bins.

Let us consider first the case 1. Let $\mathcal{O}$ be an optimum packing that has a bin containing a largest possible intersection with the set R. We claim that the packing $\mathcal{O}$ has a bin containing all squares in R. This follows from the fact that the bins in R are the largest ones in the list L_2. In fact, suppose that in the packing $\mathcal{O}$ the bin containing q_{p+1}, say Q', does not contain all the other squares in R. In this case we can obtain another packing by permuting the squares packed in Q' (that are not in R) with the squares in R, so that Q' ends up containing all the 4 squares in R, a contradiction.

Consider now the case 2. In this case, let $\mathcal{O}$ be an optimum packing that has a bin containing a largest possible intersection with the set $\{q_1\} \cup R$. According to our assumption, Q_1 is the bin containing q_1. Since the square q_{p+1} can be packed in Q_1, then it is possible to pack all the squares of R in Q_1 (together with q_1). Hence, by an appropriate permutation (if needed) we can obtain an optimum packing where the bin Q_1 contains q_1 and all the other 3 squares of R.

Consider the mentioned optimum packing $\mathcal{O}$ of L (that uses less than t bins). In case 1, respectively case 2, deleting from this packing the bin Q', respectively the bin Q_1, we obtain a packing of the corresponding list L' that uses less than $t-1$ bins. This is a contradiction to the fact that the algorithm FFDS generates a packing of L' that uses $t-1$ bins, which by induction hypothesis is optimum.

Hence, we can conclude that any optimum packing of L uses t bins, and therefore, the packing generated by the algorithm FFDS is an optimum packing of L, solving SPP(3).

The algorithm FFDS is clearly polynomial. It can be implemented to run in time $\mathcal{O}(n \log n)$. $\square$

A possible direction to extend the results presented in this section would be to study the problem SSP(k), for $k \geq 4$. We could neither prove that this problem is difficult nor obtain a polynomial time algorithm to solve it. We did not investigate this problem hard enough, as our effort was directed in the design of an approximation algorithm for SPP, but it seems to us that SPP(4) might already be $\mathcal{NP}$-hard.

4 Approximation algorithm for SPP

The negative result in Theorem 2.2 about the non-approximability of SPP—in the absolute sense—justifies the design of an approximation algorithm for this problem for which a good asymptotic performance bound can be guaranteed. In fact, asymptotic performance bounds seem to be more appropriate bounds to measure the performance of algorithms for packing problems. For the one-dimensional bin packing problem, which is not (absolutely) approximable within 1.5, it is known that the algorithm FFD (First Fit Decreasing) has this absolute performance bound (see Simchi-Levi [23]). Johnson [14] has shown that the asymptotic performance bound of this algorithm is $\frac{11}{9}$. One can find in the literature many fast algorithms for the one-dimensional bin packing problem [14, 15, 16] with asymptotic performance bound close to 1. Moreover, there exists an asymptotic polynomial approximation scheme for the one-dimensional bin packing problem, *i.e.*, an algorithm that can have an asymptotic performance bound as close to 1 as desired [8].

With this motivation, we present an approximation algorithm for SPP and show that its asymptotic performance bound is less than 1.988. We recall that it is known in the literature an algorithm with asymptotic performance bound 2.125, given by Chung, Garey and Johnson [2] for the more general problem of rectangles packing (which also solves SPP).

First we introduce some notation. We say that a square q is of type G, M or P if $s(q) \in \left(\frac{1}{2}, 1\right]$, $s(q) \in \left(\frac{1}{3}, \frac{1}{2}\right]$ or $s(q) \in \left(\frac{1}{4}, \frac{1}{3}\right]$, respectively. Given a list of squares L, we denote by $S(L)$ the total area of the squares in L. If a list L of squares is packed in a bin B, then we may refer to B as the list L. So, $S(B)$ is the total area of the squares packed in B. Given a packing $\mathcal{P}$, we denote by $\#(\mathcal{P})$ the number of bins used in the packing $\mathcal{P}$.

Let us call *area guarantee* of a bin, the minimum area occupied by the squares packed in this bin. For simplicity, we may also say area guarantee of a configuration or of a packing, referring to the area guarantee of the bin with such a configuration, or of the bins in the corresponding packing. For example, the area guarantee of configuration $C1$ of Lemma 3.1 is at least $\frac{1}{4} + 3 \cdot \frac{1}{9} = \frac{7}{12} = 0.583\ldots$.

In our algorithm we use the algorithm NFDH (Next Fit Decreasing Height) as a subroutine to pack a list of squares L into rectangular bins (a, b). The algorithm NFDH first sorts the squares in L in decreasing order of their sizes. Then packs these squares following this ordering. The squares are packed in layers, one after the other, each of width a and height defined by the first square packed in this layer. All squares are packed leftmost in each layer. When a square cannot be packed in the current layer, then it is packed in a new layer, which becomes the current layer, starting immediately after the previous layer. When a layer cannot be created in the current bin, then it is created in a new bin, which becomes the current bin. This algorithm is also denoted by NFDH^x, as it generates layers in the x-direction. There is another variant of it, denoted by NFDH^y, that generates layers in the y-direction.

The following result, due to Meir and Moser [19], is valid for the algorithm NFDH, and will be used in the analysis of the algorithm we propose.

Theorem 4.1. *Any list of squares $L = (q_1, q_2, \ldots, q_n)$ with total area A can be packed in a rectangle (a, b) by the* NFDH *algorithm if $m \geq 2$ and $s(q_i) \leq \frac{a}{m}$, $s(q_i) \leq \frac{b}{m}$ and $A \leq ab\left(\frac{1}{m^2} + \left(1 - \frac{1}{m}\right)^2\right)$.*

We are now ready to describe the algoritm we have designed for SPP, called ASP (see next page).

This algorithm generates a packing consisting of two partial packings: an optimum partial packing, $\mathcal{P}_1$, and another one with a good area guarantee, $\mathcal{P}_2$. The idea behind this algorithm is to consider sublists of L to obtain certain partial packings and then improve them, if possible. We use particular packings presented in [7], and the approach of *critical set combination* technique used in [20, 22, 21].

Let us explain the idea of critical sets used in this algorithm, and add some comments to help understand it. Note that, initially a sublist L_1' of L is considered and the algorithm FFDS is used to pack the squares in L_1', finding an optimum packing of this sublist.

For the bins used in the algorithm FFDS with the configurations given in Lemma 3.1, we have the following area guarantee.

$C1$ has area guarantee of at least $\frac{1}{4} + 3 \cdot \frac{1}{9} = \frac{7}{12} = 0.583\ldots$

$C2$ has area guarantee of at least $\frac{1}{4} = 0.25$

$C3$ has area guarantee of at least $4 \cdot \frac{1}{9} = 0.444\ldots$

Therefore, the smallest area guarantee for the bins used in the packing of L_1' is $\frac{1}{4}$ (for bins with configuration $C2$). Here, we are excluding the bin not having one of the configurations $C1$, $C2$ or $C3$ (it may exist at most one).

Algorithm ASP

Input: List of squares $L = (q_1, \ldots, q_n)$.
Output: Packing of L into unit bins.

1. Let $L_1' \leftarrow \{q \in L : \frac{1}{3} < s(q) \leq 1\}$.

2. Let $\mathcal{P}_1' \leftarrow \text{FFDS}(L_1')$.

3. Let $\mathcal{P}_A$ be the set of bins $B \in \mathcal{P}_1'$ having configuration $C2$ with a square $q \in B$, $s(q) \leq \frac{2}{3}$. (Let L_A be the set of squares in these bins.)

4. Let $L_B \leftarrow \{q \in L : 0 < s(q) \leq \frac{1}{3}\}$.

5. Let $\mathcal{P}_{AB}$ be the packing generated by filling up the bins in $\mathcal{P}_A$ as follows

5.1 Let $\mathcal{P}_{AB} \leftarrow \emptyset$.

5.2 For each bin $B \in \mathcal{P}_A$ do

5.2.1 If there exists squares in L_B then
Let $q \in B$.
Let P' be a rectangular bin of size $(1 - s(q),\ 1)$ in B (see Figure 3).
– Use the algorithm NFDH^y (starting at the bottom rightmost corner) to pack in P' as many squares of L_B as possible. Update the list L_B.
Let P'' be the largest rectangular bin $(p,\ 1 - s(q))$ in B that can be defined that does not overlap the squares packed in P'.
– Use the algorithm NFDH^x (starting at the bottom leftmost corner) to pack in P'' as many squares of L_B as possible. Update the list L_B.
$\mathcal{P}_{AB} \leftarrow \mathcal{P}_{AB} \cup B$.

6. Let L_1 be the set of all squares already packed, and $\mathcal{P}_1$ the packing generated for L_1.

7. Pack the remaining squares using the algorithm NFDH, and call $\mathcal{P}_2$ the packing that is generated.

8. Return the packing $\mathcal{P}_1 \cup \mathcal{P}_2$.

end algorithm

The algorithm first finds an optimum packing $\mathcal{P}_1'$ of L_1'. Then, it considers as *critical bins* those bins of $\mathcal{P}_1'$ with area guarantee smaller than $\frac{4}{9}$. Note that all such bins have configuration $C2$. The set of squares packed in these bins, called L_A, is then combined with the set of squares not in L_1', called L_B. The sets L_A and L_B are considered *critical.* These sets are precisely the following (see Figure 2):

$$\begin{aligned} L_A &:= \{q \in L : \frac{1}{2} < s(q) \leq \frac{2}{3}\} \text{ and} \\ L_B &:= \{q \in L : 0 < s(q) \leq \frac{1}{3}\}. \end{aligned}$$

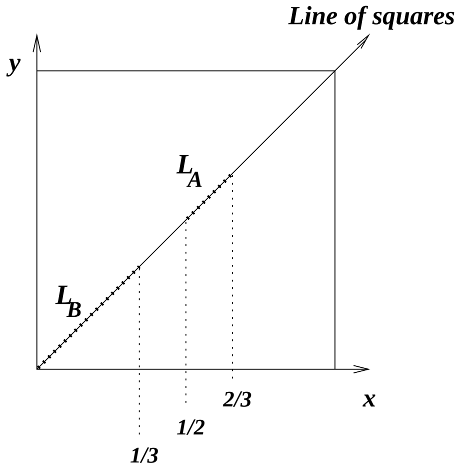

Figure 2: Critical squares in lists L_A and L_B.

The combination of the squares in the sets L_A and L_B is performed in step 5. The non-occupied region of the critical bins of $\mathcal{P}'_1$ are filled up with squares of L_B. The idea is to improve the area guarantee of these critical bins to at least $\frac{4}{9}$, or verify that all squares of L_B can be packed into these critical bins. (Note that each square of type G that is not in L_A has an area guarantee of at least $\frac{4}{9}$.)

More precisely, the combination is performed as follows. While possible, using the algorithm NFDH, the squares in L_B are packed in the non-occupied region of a critical bin in $\mathcal{P}_A$. For that, for each critical bin, first a rectangular region P' is defined (see Figure 3) and the algorithm NFDH^y is applied (starting at the bottom rightmost corner) generating levels in the y-direction. Then, in this same bin, a largest possible region $P'' = (p,\ 1 - s(q))$ that does not overlap the squares packed in P' is defined, and the algorithm NFDH^x is applied (generating levels in the x-direction). This process is applied to every critical bin in $\mathcal{P}_A$ until all critical bins in $\mathcal{P}_A$ have been considered or when all squares in L_B have been packed. The packing generated in step 5 is called $\mathcal{P}_{AB}$.

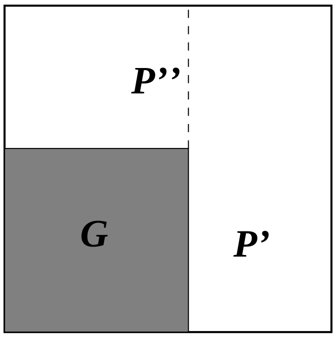

Figure 3: Small bins corresponding to P' and P''.

The final steps 6, 7 and 8 of the algorithm ASP need no further explanation.

We note that the algorithm ASP is polynomial, and can be implemented to run in time $\mathcal{O}(n \log n)$. It has asymptotic performance bound 1.988, as we show in the next theorem.

Theorem 4.2. *For any instance L of* SPP, *the following holds:*

$$\mathrm{ASP}(L) \leq 1.988 \cdot \mathrm{OPT}(L) + 7.$$

Proof. Let L be an instance for the algorithm ASP. Let L'_1 be the sublist of L defined in step 1 and $\mathcal{P}'_1$ its corresponding packing generated by the algorithm FFDS. Note that L'_1 is an instance of SPP(3). By Theorem 3.2, the algorithm FFDS finds an optimum packing of L'_1.

For the bins used in the algorithm FFDS with the configurations given in Lemma 3.1, we have seen that the smallest area guarantee is $\frac{1}{4}$ (for bins with configuration $C2$).

We recall that the *critical bins* of $\mathcal{P}'_1$ are the bins of this packing with area guarantee smaller than $\frac{4}{9}$. The set of squares in these bins is called L_A ($L_A := \{q \in L : \frac{1}{2} < s(q) \leq \frac{2}{3}\}$). These bins are filled up with squares not in L'_1, defined as L_B, until possibly all squares of L_B have been packed ($L_B := \{q \in L : 0 < s(q) \leq \frac{1}{3}\}$). The packing $\mathcal{P}_{AB}$ that is generated uses the algorithm NFDH.

We have two cases to analyse:

Case 1: All squares of L_B are packed in $\mathcal{P}_{AB}$.

In this case we obtain a final packing of L containing the same number of bins used in the packing $\mathcal{P}'_1$ of L_1. As $\mathcal{P}'_1$ is an optimum packing of L'_1 and $L'_1 \subseteq L$, we also have an optimum packing of L.

Case 2: There exist squares of L_B not packed in $\mathcal{P}_{AB}$.

Let L_2 be the set of squares in L_B not packed in $\mathcal{P}_{AB}$, and let $\mathcal{P}_2$ be the packing of L_2 generated by applying the algorithm NFDH (step 7). We analyse later the area guarantee of the packing $\mathcal{P}_2$.

First, let us analyse the area guarantee of the packing $\mathcal{P}_{AB}$. For that, we partition $\mathcal{P}_{AB}$ into 4 parts $\mathcal{P}_{AB,i}$, $i = 3, \ldots, 6$. Let $L_{B,i} := \{q \in L_B : \frac{1}{i+1} < s(q) \leq \frac{1}{i}\}$, for $i = 3, \ldots, 5$ and $L_{B,6} := \{q \in L_B : 0 < s(q) \leq \frac{1}{6}\}$. Let $\mathcal{P}_{AB,i}$ be the set of bins of $\mathcal{P}_{AB}$, with one square of L_A and at least one square of $L_{B,i}$. In Figure 4 we have some examples of bins in a packing combining squares of L_A with L_B.

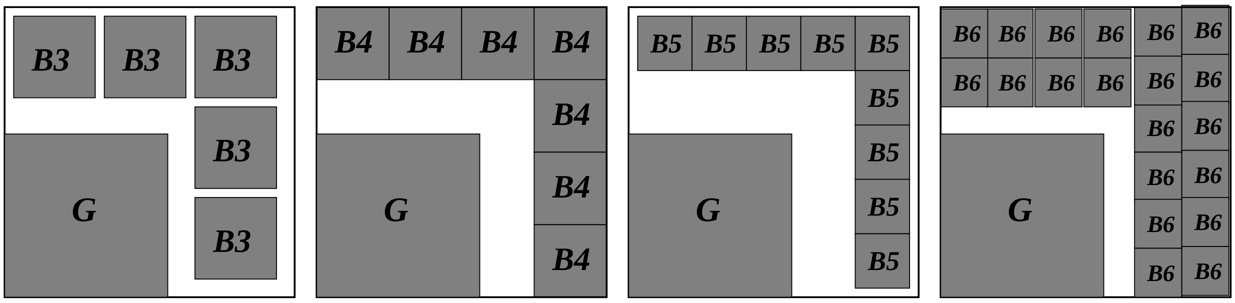

Figure 4: Combining squares of L_A and L_B.

Now we analyse the area guarantee of each packing $\mathcal{P}_{AB,i}$, $i = 3, \ldots, 6$.

1. In the packing $\mathcal{P}_{AB,3}$ we have bins with one square of L_A and 5 squares of $L_{B,3}$. Thus, this packing has an area guarantee that is at least $\frac{1}{4} + 5 \cdot \frac{1}{16} = \frac{9}{16} = 0.5625$.

2. It is immediate that the packing $\mathcal{P}_{AB,4}$ has an area guarantee that is at least $\frac{1}{4}+7\cdot\frac{1}{25} = 0.53$.
3. The packing $\mathcal{P}_{AB,5}$ has an area guarantee that is at least $\frac{1}{4} + 9 \cdot \frac{1}{36} = 0.5$.
4. Now to obtain an area guarantee for the packing $\mathcal{P}_{AB,6}$ we use Theorem 4.1.

   For that, consider the packings generated for the regions P' and P''. Note that the packing of $L_{B,6}$ into bins P' and P'' satisfies Theorem 4.1 with $m = 2$. For P' take $a = \frac{1}{3}$, $b = 1$, and note that we can take for A the value $ab\left(\frac{1}{m^2} + \left(1 - \frac{1}{m}\right)^2\right) - \left(\frac{1}{6}\right)^2$. In fact, it suffices to take as many squares of size at most $\frac{1}{6}$ as possible, while their area is at most $ab\left(\frac{1}{m^2} + \left(1 - \frac{1}{m}\right)^2\right)$. Therefore, the packing for the region P' has an area guarantee s' such that

$$\begin{aligned} s' &\geq \frac{1}{3}\cdot 1\left(\frac{1}{2^2} + \left(1 - \frac{1}{2}\right)^2\right) - \left(\frac{1}{6}\right)^2 \\ &= \frac{5}{36}. \end{aligned}$$

   Analogously, applying Theorem 4.1 we can conclude that the packing for the region P'' has an area guarantee s'' such that

$$\begin{aligned} s'' &\geq \frac{1}{3}\cdot\frac{2}{3}\left(\frac{1}{2^2} + \left(1 - \frac{1}{2}\right)^2\right) - \left(\frac{1}{6}\right)^2 \\ &= \frac{1}{12}. \end{aligned}$$

   Adding up the above areas, for each bin of $\mathcal{P}_{AB,6}$, except perhaps for the last one, we have an area guarantee that is at least $\frac{1}{4} + s' + s'' = \frac{17}{36} = 0.472\ldots$.

So, considering the combined packing $\mathcal{P}_{AB}$, we have for each bin, except perhaps for 4 bins, an area guarantee of at least $\frac{4}{9}$. That is,

$$\#(\mathcal{P}_{AB}) \leq \frac{S(L_{AB})}{4/9} + 4. \tag{1}$$

Note that $\mathrm{ASP}(L) = \#(\mathcal{P}_1) + \#(\mathcal{P}_2)$, where the packing $\mathcal{P}_1$ consists of bins having configurations $C1$, $C2$ or $C3$ and bins of $\mathcal{P}_{AB}$, and $\mathcal{P}_2$ is the packing generated in step 7.

Let us analyse the area guarantee we can have for each bin in $\mathcal{P}_1$. As mentioned previously, the area guarantee of the bins with configuration $C1$ or $C3$ is at least $\frac{4}{9}$. Each bin with configuration $C2$ has only one square, not critical, with size at least $\frac{2}{3}$; therefore, the area guarantee of each bin with this configuration is at least $\left(\frac{2}{3}\right)^2 = \frac{4}{9}$. Using these areas guarantee for bins with these configurations and inequality (1), we have the following inequality:

$$\#(\mathcal{P}_1) \leq \frac{S(L_1)}{4/9} + 5. \tag{2}$$

Moreover, as the number of bins used in $\mathcal{P}_1$ is the same as in the packing $\mathcal{P}'_1$, we know that $\mathcal{P}_1$ is an optimum partial packing and thus

$$\begin{aligned} \#(\mathcal{P}_1) &= \mathrm{OPT}(L_1) && (3) \\ &\leq \mathrm{OPT}(L). && (4) \end{aligned}$$

Let us now analyse the area guarantee of packing $\mathcal{P}_2$, generated in step 7. For that, we partition L_2 into two parts. Let $L_2' := \{q \in L_2 : \frac{1}{4} < s(q) \leq \frac{1}{3}\}$ and $L_2'' := \{q \in L_2 : 0 < s(q) \leq \frac{1}{4}\}$. In the case of L_2', the algorithm NFDH packs at least nine squares in each bin, except perhaps in the last. In this case, we have an area guarantee of at least $\frac{9}{16}$ for each bin. In the case of L_2'', using Theorem 4.1, we have an area guarantee of at least $\left(\frac{1}{m^2} + \left(1 - \frac{1}{m}\right)^2\right) - \frac{1}{m^2}$, with $m = 4$, for each bin, except perhaps the last bin. In this case, we also have an area guarantee of at least $\frac{9}{16}$ for each bin. Hence, the following inequality is valid.

$$\#(\mathcal{P}_2) \leq \frac{S(L_2)}{9/16} + 2. \tag{5}$$

Now, define $\mathcal{N}_1$ and $\mathcal{N}_2$ as follows:

$$\begin{aligned} \mathcal{N}_1 &:= \#(\mathcal{P}_1) - 5, & (6) \\ \mathcal{N}_2 &:= \#(\mathcal{P}_2) - 2. & (7) \end{aligned}$$

Combining the inequalities (2),(5),(6) and (7) we have

$$\begin{aligned} \frac{4}{9}\mathcal{N}_1 &\leq S(L_1), & (8) \\ \frac{9}{16}\mathcal{N}_2 &\leq S(L_2). & (9) \end{aligned}$$

As we are using unit square bins, $\mathrm{OPT}(L)$ is at least $S(L)$. Therefore,

$$\begin{aligned} \mathrm{OPT}(L) &\geq S(L) \\ &= S(L_1) + S(L_2) \\ &\geq \frac{4}{9}\mathcal{N}_1 + \frac{9}{16}\mathcal{N}_2. & (10) \end{aligned}$$

Using inequalities (3),(4) and (6) we have

$$\begin{aligned} \mathrm{OPT}(L) &\geq \mathrm{OPT}(L_1) \\ &= \#(\mathcal{P}_1) \\ &\geq \mathcal{N}_1. & (11) \end{aligned}$$

Combining inequalities (10) and (11), we have

$$\mathrm{OPT}(L) \geq \max\left\{\mathcal{N}_1,\ \frac{4}{9}\mathcal{N}_1 + \frac{9}{16}\mathcal{N}_2\right\}.$$

Since $\mathrm{ASP}(L) = \#(\mathcal{P}_1) + \#(\mathcal{P}_2) = \mathcal{N}_1 + \mathcal{N}_2 + 7$, we have

$$\#(\mathcal{P}) \leq \alpha \cdot \mathrm{OPT}(L) + 7,$$

where $\alpha = (\mathcal{N}_1 + \mathcal{N}_2)/(\max\{\mathcal{N}_1, \frac{4}{9}\mathcal{N}_1 + \frac{9}{16}\mathcal{N}_2\})$.

Analysing the value of α, considering each case its denominator attains the maximum, we can conclude that α is at most 1.988.

The conclusions obtained in cases 1 and 2 show that the theorem holds. $\square$

5 Concluding remarks

We have proved that there is no polynomial time algorithm for SPP with absolute performance bound less than 2 (unless $\mathcal{P} = \mathcal{NP}$). This fact motivated us to find a polynomial algorithm for this problem with asymptotic performance bound less than 2. We have described such an algorithm, called ASP, with bound 1.988.

An interesting question is whether this bound we have shown for the algorithm ASP can be improved, or maybe shown to be sharp. Other possibility would be to design variants of this algorithm using different critical sets.

It would also be interesting to find an asymptotic approximation scheme for SPP or for SPP(k), for fixed k, $k > 3$ (if these problems are not polynomially solvable).

We have described a polynomial time algorithm for SPP(3), and used it in the algorithm ASP. If a polynomial time algorithm could be designed for SPP(4), then such an algorithm could be used to design an approximation algorithm for SPP using the ideas we have used in the algorithm ASP. We do not know the computational complexity of SSP(4). It would be nice to settle this.

Further direction for research would be to investigate the problem of packing cubes into cubes, using ideas similar to those we have used here. In fact, we are carrying on some studies in this direction, but we have only some preliminary results.

References

[1] B. S. Baker, A. R. Calderbank, E. G. Coffman Jr., and J. C. Lagarias. Approximation algorithms for maximizing the number of squares packed into a rectangle. *SIAM J. Algebraic Discrete Methods*, 4(3):383–397, 1983.

[2] F. R. K. Chung, M. R. Garey, and D. S. Johnson. On packing two-dimensional bins. *SIAM J. Algebraic Discrete Methods*, 3:66–76, 1982.

[3] E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson. Approximation algorithms for bin packing - a survey. In D. Hochbaum, editor, *Approximation algorithms for NP-hard problems*. PWS, 1997.

[4] E. G. Coffman, Jr., M. R. Garey, D. S. Johnson, and R. E. Tarjan. Performance bounds for level oriented two-dimensional packing algorithms. *SIAM J. Comput.*, 9:808–826, 1980.

[5] E. G. Coffman Jr. and J. C. Lagarias. Algorithms for packing squares: A probabilistic analysis. *SIAM J. Comput.*, 18(1):166–185, 1989.

[6] E. G. Coffman, Jr. and P. W. Shor. Average-case analysis of cutting and packing in two dimensions. *European J. Operational Research*, 44:134–144, 1990.

[7] D. Coppersmith and P. Raghavan. Multidimensional on-line bin packing: algorithms and worst-case analysis. *Oper. Res. Lett.*, 8(1):17–20, 1989.

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

[9] H. Dyckhoff and U Finke. *Cutting and Packing in Production and Distribution*. Physica-Verlag, Heidelberg, 1992.

[10] H. Dyckhoff, G. Scheithauer, and J. Terno. Cutting and packing. In F. Maffioli M. Dell'AMico and S. Martello, editors, *Annotated Bibliographies in Combinatorial Optimization*, chapter 22, pages 393–412. John Wiley, 1997.

[11] P. Erdős and R. L. Graham. On packing squares with equal squares. *J. Combinatorial Theory Ser. A*, 19:119–123, 1975.

[12] M. Gardner. Some packing problems that cannot be solved by sitting on the suitcase. *Scientific American*, 241(4):22–26, October 1979.

[13] F. Göbel. *Packing and Covering in Combinatorics*, chapter Geometrical Packing and Covering Problems. published by Mathematical Centre, Tweede Boerhaavestraat 49, Amsterdam, 1979.

[14] D. S. Johnson. Fast algorithms for bin packing. *J. Comput. Syst. Sci.*, 8:272–314, 1974.

[15] D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey, and R. L. Graham. Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM J. Comput.*, 3:299–325, 1974.

[16] D. S. Johnson and M. R. Garey. A $\frac{71}{60}$ theorem for bin packing. *J. Complexity*, 1:65–106, 1985.

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

[18] K. Li and K-H. Cheng. A two-dimensional buddy system for dynamic resource allocation in a partitionable mesh connected system. *J. Parallel and Distributed Computing*, 12:79–83, 1991.

[19] A. Meir and L. Moser. On packing of squares and cubes. *J. Combinatorial Theory Ser. A*, 5:116–127, 1968.

[20] F. K. Miyazawa. *Algoritmos de Aproximação para Problemas de Empacotamento*. PhD thesis, Universidade de São Paulo IME-USP, São Paulo-SP, Brasil, november 1997.

[21] F. K. Miyazawa and Y. Wakabayashi. An algorithm for the three-dimensional packing problem with asymptotic performance analysis. *Algorithmica*, 18(1):122–144, 1997.

[22] F. K. Miyazawa and Y. Wakabayashi. Approximation algorithms for the orthogonal z-oriented three-dimensional packing problem. *SIAM J. Comput.*, to appear.

[23] D. Simchi-Levi. New worst-case results for the bin-packing problem. *Naval Res. Logistics*, 41:579–585, 1994.