Approximation Schemes for Multidimensional Packing

José R. Correa * Claire Kenyon †

Abstract

We consider a classic multidimensional generalization of the bin packing problem, namely, packing $d-$dimensional rectangles into the minimum number of unit cubes. Our two results are: an asymptotic polynomial time approximation scheme for packing d-dimensional cubes into the minimum number of unit cubes and a polynomial time algorithm for packing rectangles into at most OPT bins whose sides have length $(1+\epsilon)$, where OPT denotes the minimum number of unit bins required to pack the rectangles. Both algorithms also achieve the best possible additive constant term. For cubes, this settles the approximability of the problem and represents a significant improvement over the previous best known asymptotic approximation factor of $2-(2/3)^d+\epsilon$. For rectangles, this contrasts with the currently best known approximation factor of $1.691\ldots$.

1 Introduction

Bin packing problems have attracted much attention in the literature since the seventies. In the one dimensional case the problem consists of using the minimum number of unit (one dimensional) bins to pack a list of items of size at most one. One dimensional bin packing is known to be NP-hard; moreover, no approximation algorithm with approximation factor better than 3/2 can exist unless $P=NP$. This leads to the consideration of asymptotic performance guarantees: Fernandez de la Vega and Lueker [5] thus settled the approximability of one dimensional bin packing by giving an asymptotic approximation scheme. In spite of that, in many real world applications the one dimensional framework is too limited and extensions to higher dimensions are needed. Unfortunately, the treatment of higher dimensional versions of bin packing has not been as satisfactory as the one dimensional case so far.

Several generalizations of one dimensional bin packing have been considered. Square packing, vector packing and rectangle packing seem to be the most commonly known extensions. The square packing problem consists of packing a list of squares into the minimum number of unit squares (bins). Leung et al. [10] proved that deciding whether a list of squares can be packed in a unit square is NP-hard; therefore, square packing is NP-hard. This immediately implies [6] that there is no better than 2 approximation algorithm for square packing unless $P=NP$. On the other hand, the best known asymptotic approximation algorithms for square packing, independently obtained by Seiden and van Stee [13] and Kohayakawa et al. [9], have an asymptotic performance guarantee of $14/9+\epsilon$ for any $\epsilon>0$. Kohayakawa et al.'s algorithm also works in higher dimensions and has an asymptotic approximation factor arbitrarily close to $2-(2/3)^d$; furthermore, it runs in polynomial time when d is a fixed constant.

Here we first study the $d-$dimensional cube packing problem, d-CP for short. For a list I of $d-$dimensional cubes, let us denote by $\mathrm{OPT}(I)$ the minimum number of unit cubes in which I can be packed. The first main result in this paper reads as follows.

THEOREM 1.1. *There exists an algorithm A which, given a list I of n $d-$dimensional cubes and a positive number ϵ, produces a packing of I into $A(I)$ copies of $[0,1]^d$ such that:*

$$A(I) \leq (1+\epsilon)\ OPT(I) + 1.$$

The running time of A is polynomial for fixed d and ϵ.

Note that our approximation scheme is best possible in the sense that the hardness result above implies that there can not be an approximation scheme with constant term strictly less than 1. (Nevertheless, there is not enough evidence to rule out the existence of a polynomial time algorithm producing a packing into no more than $\mathrm{OPT}(I)+1$ bins.). Theorem 1.1 was independently proved by Bansal and Sviridenko [1] but with an additive constant of $O(1)$, that depends on $1/\epsilon$, instead of just 1.

The rectangle packing problem (or two dimensional bin packing) consists of packing a list of rectangles into the minimum number of unit squares. Recently, Bansal and Sviridenko [1] showed the following very interesting result: there is no asymptotic polynomial

*Operations Research Center, Massachusetts Institute of Technology, Cambridge, MA 02139-4307. jcorrea@mit.edu

†Laboratoire d'Informatique, Ecole Polytechnique, 91128 Palaiseau Cedex. kenyon@lix.polytechnique.fr

time approximation scheme for rectangle packing unless $P = NP$. Their strong result contrasts with the best approximation algorithm has an asymptotic guarantee of 1.691... obtained by Caprara [3]. Nevertheless, the approximation gap is still wide. Here is our second main result.

THEOREM 1.2. *There exists an algorithm A which, given a list I of rectangles and a positive number ϵ, produces a packing of I into $A(I)$ copies of $[0, 1+\epsilon]^2$ such that:*

$$A(I) \leq \ OPT(I),$$

where $OPT(I)$ is the minimum number of unit cubes in which I can be packed. The running time of A is polynomial for fixed ϵ.

The last theorem is also best possible in the sense that the constant cannot be improved and that there is no APTAS for rectangle packing.

In Section 2, we prove Theorem 1.1. Besides using extensions of the approach of Fernandez de la Vega and Lueker [5], the main idea behind the hypercube packing algorithm is to divide the input list into three sets, "large", "medium" and "small", in such a way such that the medium cubes are negligible. As a warm up, we first see how an almost optimal packing of the small cubes can be found, then we do the same for large cubes; finally we bring the pieces together and show how the appropriate partition is found.

In Section 3, we prove Theorem 1.2. There is no natural total order on rectangles, hence the rounding technique used by Fernandez de la Vega and Lueker does not seem to extend here. Hence, we round sizes in a straightforward manner, at the cost of enlarging the bin sizes from $[0, 1]^2$ to $[0, 1+\epsilon]^2$. This enables one to deal with rectangles which are both wide and tall. The main problem is to deal with the rectangles which are either very wide and flat or very thin and tall, and for either kind we can essentially use the strip-packing algorithm of Kenyon and Rémila [7]. Our algorithm thus relies on an appropriate partition of rectangles into "large", "small", "horizontal", "vertical", and "medium", in such a way that the medium rectangles are negligible. It only remains to mix gracefully the various kinds of rectangles; this requires discretizing a near-optimal solution appropriately so as to be able to "guess" not only where the large rectangles go, but also the areas used to pack horizontal rectangles and the areas used to pack vertical rectangles.

Finally, in Section 4, we discuss a related packing problem with applications in VLSI design. Specifically, we show how the ideas in Section 3 can be used to obtain a PTAS for this problem.

2 Hypercube Packings

2.1 Definitions and Preliminaries. We adopt two standard assumptions in multidimensional bin packing: first, items are allowed to "touch" i.e., they can intersect in a face; second, items can not be rotated (*orthogonal packing without rotation*).

A $d-$dimensional cube is given by a positive number a, representing the length of its side. Since we will pack squares into unit squares, we need to assume $a \leq 1$. Given an input list I of n $d-$dimensional cubes with sides of length $a_i \in [0, 1]$ for $i = 1, , \ldots, n$, the volume of the list is defined as: $vol(I) = \sum_{i=1}^{n} a_i^d$. Given two cubes and their positions in the space, we say that they are *nonoverlaping* if their interiors are disjoint.

A packing of I into k bins, is a positioning of the cubes into k copies $H_1, \ldots, H_k$ of the unit hypercube $[0, 1]^d$, so that no two cubes overlap. The $d-$dimensional cube packing problem consists of finding a packing of I into the minimum number of bins, $\mathrm{OPT}(I)$. We denote $A(I)$ the number of bins algorithm A uses to pack I.

We now establish a simple result that will be needed later.

LEMMA 2.1. *Let S be a set of n nonoverlaping $d-$dimensional cubes positioned inside $[0, 1]^d$. Then, $[0, 1]^d \setminus S$ can be seen as the union of no more than $(2n + 1)^d$ nonoverlaping $d-$dimensional rectangles.*

Proof. Extending each facet of each cube in S, we obtain a grid in $[0, 1]^d$ consisting of $(2n + 1)^d$ cells. ∎

OBSERVATION 2.1. *In the last lemma we can assume without loss of generality that in the set S there is a cube touching each of the hyperplanes $x_i = 0$, for $i = 1, \ldots, d$. Thus, $[0, 1]^d \setminus S$ can be seen as the union of no more than $(2n)^d$ nonoverlaping $d-$dimensional rectangles.*

OBSERVATION 2.2. *In the 2-dimensional case, no more than $3n$ rectangles are needed to cover $[0, 1]^2 \setminus S$. This bound can be seen as follows. For each square in S draw an horizontal line at its top and bottom until it intersects some other square as in Figure 1. Assuming the bottom-most square is at height 0, we have drawn $2n - 1$ horizontal lines. This partitions the unit square into at most $3n$ rectangles (two to the side of each square and another to the top of each square in S) plus the original n squares.*

2.2 Packing Small Cubes. In this section we look at multidimensional cube packing in case all cubes are small. In this setting we are given an input list S of cubes with sides $a_i \leq \delta$ for some positive (small) constant δ. We will use a multidimensional version of

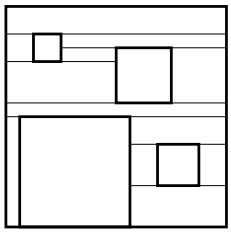

Figure 1: Decomposing the unit square into rectangles

the Next-Fit-Decreasing-Height shelf heuristic (NFDH). Coffman, Garey, Johnson, and Tarjan [2] analyzed this heuristic in the context of strip packing pointing out properties closely related to what we extend here to higher dimensions.

In two dimensions, the NFDH algorithm is a level algorithm which uses the next-fit approach to pack the sorted (by nonincreasing size) list of squares. The squares are packed, left-justified on a level (shelf) until the next rectangle will not fit. This rectangle is used to define a new level and the packing continues on this level. The earlier levels are not revisited. In general we define the NFDH heuristic inductively. Assume we know how to perform NFDH in $d-1$ dimensions. The $d-$dimensional NFDH heuristic will look at a facet $\mathcal{F}$ of the bin and pack squares on it using its $d-1$ dimensional version. After it finishes packing the facet, it will cut off from the bin a $d-$dimensional rectangle (*slice*) with base $\mathcal{F}$ and height a_1. It will proceed packing the remaining list in the rest of the bin.

LEMMA 2.2. *Let S denote the instance and assume all cubes in S have side smaller than δ. Consider the NFDH heuristic applied to cubes in S. If NFDH can not place any other cube in a rectangle R of size $r_1 \times r_2 \times \cdots \times r_d$ (with $r_i \leq 1$), the total wasted (unfilled) space in that bin is no more than:*

$$2\delta \sum_{i=1}^{d} r_i$$

Proof. The key idea of the proof is that the smallest cube in slice i is larger than the largest cube in slice $i+1$. This says that the volume covered by cubes in slice i is almost as big as (could be even bigger) the total volume of slice $i+1$.

Let us denote by $h_1, h_2, \ldots, h_l$ the height of the slices generated by the algorithm. Clearly $\sum_{i=1}^{l} h_i \geq 1-\delta$. For $d=2$ the total wasted space can be bounded by $2\delta r_1 + \delta r_2$. The factor $2\delta r_1$ comes from the first slice (that we cannot bound) and the wasted space above the last slice while the factor δr_2 arises from the wasted space to the right of each slice.

The same argument above applies in general and the total wasted space can be bounded by

$$\epsilon \cdot \prod_{j \neq 1} r_j + \sum_{i=1}^{d} \epsilon \cdot \prod_{j \neq i} r_j \leq 2\epsilon \sum_{i=1}^{d} r_i.$$

Note that, as in the two-dimensional case, only for $i=1$ we waste the volume of two slices. ∎

OBSERVATION 2.3. *If in the previous lemma we are using the NFDH heuristic to place cubes in many rectangles, a slightly more careful argument (which will bound the height of slice 1 of a given rectangle R_i with the size of the smallest item in the last slice of rectangle R_{i-1}) can be used to improve the bound on the wasted space in rectangle R to: $\delta \sum_{i=1}^{d} r_i$.*

COROLLARY 2.1. *Let S denote the instance and assume all cubes in S have side smaller than δ, then*

$$NFDH(S) \leq (1+2d\delta)vol(S) + 1.$$

That is, NFDH is an APTAS when all cubes are small.

2.3 Packing Large Cubes. We now concentrate in the case of packing large cubes. In this case we are given an input list L of n cubes whose sides are at least δ. For simplicity, we split the analysis into two steps (as in [5, 14]).

LEMMA 2.3. *If the input list L contain only large cubes, say $a_i \geq \delta$ for all $i = 1, \ldots, n$ and there are only K different cube sizes for some constant K. Then, we can solve d-CP optimally in polynomial time.*

Proof. Clearly the number of items that fit in a bin is bounded by $M = \lfloor 1/\delta^d \rfloor$. Let us see that the number of bin types R is also constant.

Assume cubes of type i have side length equal to α_i. A bin type will be denoted by a vector $(x_1, x_2, \ldots, x_K)$ i.e., x_1 cubes of type 1, x_2 cubes of type 2, up to x_K cubes of type K , that fit in a bin. If a given vector of cubes fit in a bin, they have fit in an arrangement (position) that place their vertices in points with coordinates belonging to the set:

$$\mathcal{C} = \left\{ \sum_{i=1}^{K} \lambda_i \alpha_i : \lambda_i \in \mathbb{Z}_+ \text{ and } 0 \leq \sum_{i=1}^{K} \lambda_i \alpha_i \leq 1 \right\}.$$

Indeed, given a general placement of the cubes in the bin, fix a dimension $l = 1, \ldots, d$ and sort the cubes by their leftmost point in their $l-$th coordinate. Now push the cubes one-by-one, starting from the cubes with smaller $l-$th coordinate, a little bit to the left until the

vertices' $l-$th coordinates belong to $\mathcal{C}$. By doing this for all coordinates we obtain the claimed result.

Now let C denote the cardinality of the set $\mathcal{C}$. Since $\alpha_i \geq \delta$ any point $\sum_{i=1}^{K} \lambda_i \alpha_i$ in $\mathcal{C}$ will satisfy $\lambda_i \leq 1/\delta$ for all $i = 1, \ldots, K$. Hence, $C \leq (1/\delta)^K$ which is a (huge) constant. This implies that in a bin type all cubes will be placed with their vertices in $\mathcal{C}^d$. Then, there is at most $\binom{C^d}{M} M^K$ possible bin types, which is again a constant. Moreover, we can check in constant time if a given vector $(x_1, \ldots, x_K)$ is a bin type or not.

Therefore, both M and R are constants and clearly $\mathrm{OPT}(L) \leq n$. Hence, we can try all possible packings in polynomial time, in fact no more than $\binom{n+R}{R}$ such packings can exist. ∎

LEMMA 2.4. *If $a_i \geq \delta$ for all $i = 1, \ldots, n$ then there is a $(1+\delta)$ approximation algorithm for d-CP.*

Proof. Let L denote the given instance list. Sort the n cubes in nonincreasing order of sizes and partition them in $K = 1/\delta^{d+1}$ groups, each containing at most $Q = n/K$ cubes. We construct two instances J (resp. J') by rounding up (resp. down) all items in each group to the largest (resp. smallest) cube in the group. Clearly

$$\mathrm{OPT}(J') \leq \mathrm{OPT}(L) \leq \mathrm{OPT}(J).$$

Moreover, the argument of Fernandez de la Vega and Lueker applies here as well, then,

$$\begin{aligned} \mathrm{OPT}(J) &\leq \mathrm{OPT}(J') + Q \\ &\leq \mathrm{OPT}(L) + Q \\ &\leq (1+\delta)\ \mathrm{OPT}(L). \end{aligned}$$

The last inequality follows directly from $\mathrm{OPT}(L) \geq n\delta^d$ together with $Q \leq n\delta^{d+1}$.

Therefore, the algorithm is to first round the instance and then apply the previous lemma to solve the rounded instance optimally. ∎

2.4 The General Case in Two Dimensions. In this section we prove Theorem 1.1 in the case cubes of any size are allowed to be part of the input list that we now denote by I. Although we already know how to construct almost optimal packings of only big and only small cubes separately, we can not directly apply the algorithms in the previous sections to construct a packing in the general case. One extra step will be required to allow the combination of the previously seen algorithms.

For the sake of simplicity in the notation we work in this section only in the two dimensional case. The extension to higher dimensions is straightforward and discussed at the end of the section.

The general asymptotic polynomial time approximation scheme for square packing is as follows:

(1) Let I be the input list and $\epsilon > 0$ fixed. Consider the sequence $\epsilon, \epsilon^3, \ldots, \epsilon^{2^i-1}, \ldots$ and let $M_i = \{j : a_j \in [\epsilon^{2^{i+1}-1}, \epsilon^{2^i-1})\}$.

(2) Take $M := M_i$ for some index i satisfying $vol(M_i) \leq \epsilon\ \mathrm{OPT}(I)$. Define the set of large items as $L = \{j : a_j > \epsilon^{2^i-1}\}$ and the set of small items as $S = \{j : a_j \leq \epsilon^{2^{i+1}-1}\}$.

(3) Find an almost optimal packing of L as in Lemma 2.4.

(4) Use NFDH to pack as many squares in S as possible using only the remaining space in the already opened bins (possibly a set $S' \subset S$ was not packed).

(5) Use NFDH using only new bins to pack $M \cup S'$.

From now on, we call this algorithm as algorithm A. We analyze its running time and prove that satisfies the bound in Theorem 1.1.

Analysis of the Running Time. It is clear that steps (3)-(5) take polynomial time for fixed d and ϵ. We then only need to see that in step (1)-(2) we can find the set M_i such that $vol(M_i) \leq \epsilon\ \mathrm{OPT}(L)$ in polynomial time. Indeed, since:

$$\sum_{i \leq \lceil 1/\epsilon \rceil} vol(M_i) \leq \mathrm{OPT}(I)$$

and there are $\lceil 1/\epsilon \rceil$ terms (a constant amount) in the sum. We can find an index i with the desired property by exhaustive search. Hence, algorithm A runs in polynomial time.

Analysis of the Algorithm. To prove that algorithm A satisfies the result in Theorem 1.1, we need to distinguish two cases depending on what the set S' was after step (4). In both cases the bound in the theorem holds.

(i) *After step (4), S' is empty.* In this case, after step (4) the number of bins algorithm A has opened is bounded by

$$(1+\epsilon^{2^i-1})\ \mathrm{OPT}(L \cup S) \leq (1+\epsilon)\ \mathrm{OPT}(I).$$

And it only remains to pack the squares in M. For that purpose algorithm A needs at most

$$(1+\epsilon^{2^i-1})\epsilon\ \mathrm{OPT}(I) + 1 \leq 2\epsilon\ \mathrm{OPT}(I) + 1$$

new bins. It follows that the total number of bins used by A was no more than

$$(1+3\epsilon)\ \mathrm{OPT}(I)+1.$$

(ii) *After step (4), S' is nonempty.* In this case we derive a volume argument for the bins opened to pack L. Consider a bin containing a subset L' of squares in L, clearly $|L'| \leq 1/\epsilon^{(2^i-1)2}$. From Lemma 2.1 $[0,1]^2 \setminus L'$ can be decomposed into no more than $3/\epsilon^{(2^i-1)2}$ rectangles, these are filled in step (4) of the algorithm (by NFDH) with squares from the set S. Lemma 2.2 tells us that for each rectangle in the partition we will cover everything but a volume of at most $2\epsilon^{2^{i+1}-1}$ (since each rectangle in the partition has sides length no more than 1). Adding over all rectangles, the total wasted space can be bounded by $2\epsilon^{2^{i+1}-1} \times 3/\epsilon^{(2^i-1)2} \leq 6\epsilon$. This last bound implies that for each bin containing squares in L, at least a fraction $(1-6\epsilon)$ of its volume is filled with squares from $L \cup S$.

To finish the proof note that $A(I)$ is exactly the sum of the number of bins used up to step (4) and the number of bins used in step (5). The first quantity was bounded in the last paragraph while the second can be bounded by Lemma 2.2. Thus,

$$\begin{aligned} A(I) &\leq \frac{vol(L \cup (S \setminus S'))}{1-6\epsilon} \\ &\quad +(1+2\epsilon)vol(S' \cup M)+1 \\ &\leq (1+12\epsilon)vol(I)+1 \\ &\leq (1+12\epsilon)\ \mathrm{OPT}(I)+1, \end{aligned}$$

proving Theorem 1.1 in the two dimensional case.

The Higher Dimensional Analysis. The asymptotic approximation scheme in the d dimensional case is almost the same as the one described above. The only difference lies in the sequence in step (1) and the consequent definition of L, M and S. In d dimensions steps (1) and (2) should be replaced by:

(1') Let I be the input list and $\epsilon > 0$. Consider the sequence

$$\alpha_i = \epsilon^{\frac{d^{2(i+1)}-1}{d^2-1}} \qquad \text{for } i \geq 0,$$

and let $M_i = \{j : a_j \in [\alpha_{i+1}, \alpha_i)\}$.

(2') Take $M := M_i$ for some i such that $vol(M) \leq \epsilon\ \mathrm{OPT}(I)$. Define the set of large items as $L = \{j : a_j > \alpha_i$ and the set of small items as $S = \{j : a_j \leq \alpha_{i+1}\}$.

Clearly, the running time of the algorithm is again polynomial in the input size if d and ϵ are fixed constant.

Finally, the analysis of the algorithm is exactly the same in case **(i)** and very similar in case **(ii)**. The only difference is that we will need to use the bounds from the all small and all large items in the $d-$dimensional setting. Therefore, the actual numbers we obtain will depend on d, which is a fixed constant.

3 Packing Rectangles

The approach taken above could also be attempted for rectangle packing. Unfortunately we run into trouble when rounding the large rectangles. Instead, our approach here consists in relaxing the constraints by allowing to enlarge the bins slightly. Here is the algorithm used to prove Theorem 1.2.

3.1 The Algorithm.

Input. Denote the input list by I. Assume that the ith rectangle has width a_i and height b_i, with $0 \leq a_i, b_i \leq 1$. Denote also by $\mathrm{Sf}(I) = \sum_{i=1}^{n} a_i b_i$, the total surface of the input.

Partitioning the Input. For $j \in \{1, 2, \ldots, 2/\epsilon\}$, let M_j denote those rectangles such that $a_i \in (\epsilon^{2j+1}, \epsilon^{2(j+1)+1}]$ or $b_i \in (\epsilon^{2j+1}, \epsilon^{2(j+1)+1}]$. Let i_0 be such that the total surface area of the rectangles of M_{i_0} is minimum. Let $\epsilon' = \epsilon^{2i_0+1}$, and define the partition $I = M_{i_0} \cup L \cup H \cup V \cup S$, where:

- $L = \{i : a_i > \epsilon'$ and $b_i > \epsilon'\}$
- $S = \{i : a_i < \epsilon'\epsilon^2$ and $b_i < \epsilon'\epsilon^2\}$
- $H = \{i : a_i > \epsilon'$ and $b_i < \epsilon'\epsilon^2\}$
- $V = \{i : a_i < \epsilon'\epsilon^2$ and $b_i > \epsilon'\}$

Rounding the Input. We now round every coordinate greater than ϵ' up to the nearest multiple of $\epsilon'\epsilon$. Denote by I' the rounded instance.

Let C denote the number of distinct rectangles in L: thus L contains ℓ_i rectangles of type i, for $1 \leq i \leq C$.

Rounding and Partitioning the Output. We define bin types by decomposing $(1+\epsilon) \times (1+\epsilon)$ squares as follows:

- We consider all possible packings of (large) rectangles of type i, $1 \leq i \leq C$, into a $(1+\epsilon) \times (1+\epsilon)$ square, such that the corners of the rectangles have coordinates which are integer multiples of $\epsilon'\epsilon$.
- For each possible packing into a $(1+\epsilon) \times (1+\epsilon)$ square, we consider the area of the bin which is still uncovered as a union of small square cells of side length $\epsilon'\epsilon$ (where each square is positioned at

integer multiples of $\epsilon'\epsilon$), and label each cell either H or V.

Together with this labeling of the uncovered area, this packing of large rectangles defines a bin type. Let K be the total number of bin types.

Main Loop. For each $(n_1, \ldots, n_K)$ such that $\sum_j n_j \leq n$, we attempt to construct a packing of I' using n_j bins of type j, such that the rectangles from L are packed in the spaces reserved for them in the bin type, the cells labeled H are only used for rectangles from $H \cup S$ and the cells labeled V are only used for rectangles from $V \cup S$. This is done as follows.

1. To decide whether the rectangles from L can be placed, we check that for every rectangle type i, $1 \leq i \leq C$, the number ℓ_i of type i rectangles in I' is less than or equal to the total space available for them:

$$\ell_i \leq \sum_{1 \leq j \leq K} n_j \cdot \left(\begin{array}{c} \text{number of type } i \text{ rectangles} \\ \text{positioned in type } j \text{ bins} \end{array} \right).$$

2. We use the following algorithm for packing the rectangles from H.

   (a) For each bin type j, consider the union U of the cells labeled H. Drawing horizontal lines at y-coordinates integer multiples of $\epsilon'\epsilon$, we can interpret U as a union of horizontal strips of height $\epsilon'\epsilon$ and width multiple of $\epsilon'\epsilon$. For each integer multiple ℓ of $\epsilon'\epsilon$, let $h_\ell^{(j)}$ denote the sum of the heights of the strips of width ℓ in a type j bin. Let h_ℓ denote the total height of the strips of width ℓ in the packing which we are currently constructing,

$$h_\ell = \sum_{1 \leq j \leq K} n_j h_\ell^{(j)}.$$

   (b) To pack rectangles from H, we will solve the following fractional strip-packing problem. Consider all *configurations* $(w_1, w_2, \ldots)$ of widths which are multiples of $\epsilon'\epsilon$ and sum to at most $1 + \epsilon$. Let q be the number of such configurations. Let A_{ir} denote the number of occurrences of the width $i\epsilon'\epsilon$ in configuration r. Let B_i denote the sum of all heights of the rectangles of H whose width equals $i\epsilon'\epsilon$. We define one variable $x_r^{(\ell)}$ for each strip width ℓ and for each configuration r whose widths sum to at most ℓ. We find, in polynomial time, a basic feasible solution to the following system of linear constraints, if it exists.

$$\begin{cases} (\forall i) & B_i & \leq & \sum_{\ell,r} x_r^{(\ell)} A_{ir} \\ (\forall \ell) & \sum_{r,\ell} x_r^{(\ell)} & \leq & h_\ell \\ (\forall r, \ell) & x_r^{(\ell)} & \geq & 0 \end{cases}$$

   (c) We place rectangles from H in the configurations thus defined, proceeding in a greedy fashion.

   (d) We cut back into strips of height $\epsilon'\epsilon$ and place them back into the bins.

   Let M denote the set of rectangles which: either did not fit in the fractional packing after (c) or are cut in the process (d). Now, we now set aside M. This defines a packing of the rectangles of $H \setminus M$ into the parts of the bins labeled H.

3. Similarly, we pack the rectangles of $V \setminus M'$ into the parts of the bins labeled V.

4. We pack the rectangles of S into the $\epsilon'\epsilon \times \epsilon'\epsilon$ cells which have available space, using the Next Fit Decreasing Height (NFDH) algorithm.

5. We expand each bin by adding a thin vertical $1 \times O(\epsilon)$ horizontal strip and a this $O(\epsilon) \times 1$ vertical strip and use them to pack the rectangles from $M_{i_0} \cup M \cup M'$ using an $O(1)$-approximation algorithm such as NFDH in the horizontal strips and FFDW (Width) in the vertical strips.

Output. We output the best packing among all feasible packings of L thus constructed.

3.2 Analysis of the Running Time. The running time is relatively easy to analyze. $i_0 \leq 2/\epsilon$, thus $\epsilon' \geq \epsilon^{4/\epsilon} = \Omega(1)$. The number C of large rectangle types is at most $(1/\epsilon'\epsilon)^2 = O(1)$. A bin type can be defined by labeling each $\epsilon' \times \epsilon$ cell by H, V, or $i \leq C$, thus the number K of bin types is at most $(C+2)^{1/(\epsilon'\epsilon)^2} = O(1)$, and so the number of iterations through the main loop is at most n^K which is polynomial in n.

The number of strip widths is at most $1/\epsilon'\epsilon = O(1)$. The number of configurations is at most $2^{2/(\epsilon'\epsilon)} = O(1)$. Multiplying, the number of variables in the linear program is $O(1)$. The number of constraints is at most $2/(\epsilon'\epsilon) = O(1)$. The coefficients $A_{i,r}$ are bounded by $O(1)$ and B_i are written on at most $O(\log n)$ bits, hence the linear program can be solved efficiently.

The First Fit Decreasing Height and First Fit Decreasing Width algorithms are polynomial time.

Overall, the algorithm thus runs in polynomial time.

3.3 Analysis of Correctness.

LEMMA 3.1.

$$Sf(M_{i_0}) \leq \epsilon Sf(I).$$

Proof. Each rectangle of I belongs to at most two sets M_j, thus $\sum_{1\leq j\leq 2/\epsilon} \mathrm{Sf}(M_j) \leq 2\mathrm{Sf}(I)$. The minimum surface is less than the average surface, which is bounded by $\epsilon\mathrm{Sf}(I)$. ∎

LEMMA 3.2. *Let I' denote the rounded input. If I can be packed into OPT unit size bins, then I' can be packed into OPT $(1+2\epsilon)\times(1+2\epsilon)$ bins in such a way that any rectangle with $a'_i \geq \epsilon'$ is positioned at an x-coordinate which is an integer multiple of $\epsilon'\epsilon$, and any rectangle with $b'_i \geq \epsilon'$ is positioned at a y-coordinate which is an integer multiple of $\epsilon'\epsilon$.*

Proof. Consider the optimal packing of I. Define a partial order $\prec_H$ on rectangles as the transitive closure of the relation: i is in relation with i' if rectangle i, when translated horizontally to the right by $2\epsilon'\epsilon$, intersects i'. Note that any chain in $\prec_H$ contains at most $1/\epsilon'$ rectangles with $a_i \geq \epsilon'$.

Take a total order $\leq$ which extends $\prec_H$. We take the rectangles one by one in that order and deal with i as follows: we extend i horizontally to the right so that its width becomes a'_i, we translate it horizontally to the right to that it is positioned at an integer multiple of $\epsilon'\epsilon$; then, for each $i' \geq i$ in increasing order, if i' intersects some rectangle $i'' \leq i'$ then we translate i' to the right by $2\epsilon'\epsilon$.

It is easy to check that this constructs a feasible packing.

Since the chains in $\prec_H$ contain at most $1/\epsilon'$ rectangles of width $a_i \geq \epsilon'$, each rectangle is translated at most $1/\epsilon'$ times, hence in total by at most $(2\epsilon'\epsilon)/\epsilon' = 2\epsilon$. Thus the final packing fits in $(1+2\epsilon)\times 1$ bins.

We then proceed similarly for rounding the b_is into b'_i. ∎

LEMMA 3.3. *Consider a packing of I' into $(1+2\epsilon)\times(1+2\epsilon)$ bins, satisfying the condition of Lemma 3.2. Consider any cell $\mathcal{C} = [m\epsilon'\epsilon, (m+1)\epsilon'\epsilon]\times[p\epsilon'\epsilon, (p+1)\epsilon'\epsilon]$ in any bin. Then either $H\cap\mathcal{C} = \emptyset$ or $V\cap\mathcal{C} = \emptyset$.*

Proof. Assume, for a contradiction, that $i \in H\cap\mathcal{C} \neq \emptyset$ and $i' \in V\cap\mathcal{C} \neq \emptyset$. Since i has width and starting point integer multiples of $\epsilon'\epsilon$, it must be that i spans the whole width of $\mathcal{C}$. Similarly, i' must span the whole height of $\mathcal{C}$, But then i and i' must intersect, a contradiction. ∎

LEMMA 3.4.

$$Sf(M) = O(\epsilon)Sf(I)$$

and

$$Sf(M') = O(\epsilon)Sf(I).$$

Proof. Let us only prove the first equation, the second is analogous. Consider the sets M_c and M_d denoting the rectangles the were discarded in steps (c) and (d) respectively. Then $M_c \cup M_d = M$. We prove that both $\mathrm{Sf}(M_c) = O(\epsilon)\mathrm{Sf}(I)$ and $\mathrm{Sf}(M_d) = O(\epsilon)\mathrm{Sf}(I)$.

To see the first equality, we recall the strip packing analysis by Kenyon and Remila [7]. Consider a fractional strip packing (i.e. a solution the to linear program in step (2)(b)) $x_r^{(\ell)}$ for all r and ℓ. Clearly such solution has at most $(2/\epsilon'\epsilon)$ nonzero coordinates. Fix ℓ and let $x_1^{(\ell)}, \ldots, x_k^{(\ell)}$ be the nonzero variables corresponding to that ℓ. To constrict an integer strip packing we proceed as follows: Let $x_j^{(\ell)} > 0$ be the variable corresponding to the current configuration. This configuration will be used between levels $l_j^\ell = (x_1^{(\ell)} + \epsilon'\epsilon^2) + \cdots + (x_{j-1}^{(\ell)} + \epsilon'\epsilon^2)$ and $l_{j+1}^\ell = l_j^\ell + x_j^{(\ell)} + \epsilon'\epsilon^2$. For each i such that $A_{ij} \neq 0$ we draw A_{ij} columns of width $i\epsilon'\epsilon$ going from level l_j^ℓ to level l_{j+1}^ℓ. After this is done for all ℓ and all configurations, we take all columns of width $i\epsilon'\epsilon$ and start filling them up with the corresponding rectangles of the same width in a greedy manner. It is not difficult to show that all rectangles in fit. Now, we take all rectangles whose top end belongs to the interval $(l_j^\ell - \epsilon'\epsilon^2, l_j^\ell]$, for any ℓ and j, and set them aside (they are put in M_c). The total surface of the removed rectangles is clearly no more than

$$\begin{aligned} &2 \cdot (\text{ number of constraints }) \cdot (\text{ max item height }) \\ &= 2 \cdot \frac{2}{\epsilon'\epsilon} \cdot \epsilon'\epsilon^2 = 4\epsilon. \end{aligned}$$

This proves that a fractional strip packing where strips of width ℓ are of height h_ℓ (the linear program in step (2)(b)), can be turned into an integer strip packing of almost all rectangles where strips of width ℓ are of height no more than h_ℓ.

Therefore, the total surface of rectangles in H that may not fit after step (2)(c) is bounded by $2(2/\epsilon'\epsilon)\epsilon'\epsilon^2 = 4\epsilon$. In other words $\mathrm{Sf}(M_c) = O(\epsilon)\mathrm{Sf}(I)$.

On the other hand, to see that $\mathrm{Sf}(M_d) = O(\epsilon)\mathrm{Sf}(I)$. Notice that the rectangles in H have height smaller than $\epsilon'\epsilon^2$ and the cut strips of step (2)(d) are of height $\epsilon'\epsilon$, the surface of the rectangles put aside in step (2)(d) is no more than a fraction ϵ of the surface of H.

Hence, he total surface of M is no more than a fraction $O(\epsilon)$ of the total surface of the instance. ∎

LEMMA 3.5. *The rectangles in $M_{i_0} \cup M \cup M'$ fit into the thin strips added along the bins in step (5).*

Proof. Clearly the whole surface of $M_{i_0} \cup M \cup M'$ is no more than $O(\epsilon)\text{Sf}(I)$. Moreover, we can partition $M_{i_0} \cup M \cup M'$ into two sets A and B such that:

- A contains only rectangles with $a_i < \epsilon'\epsilon^2 < \epsilon^2$ and $\text{Sf}(A) = O(\epsilon)\ \text{OPT}(I)$ (A contains M' and part of M_{i_0}).
- B contains only rectangles with $b_i < \epsilon'\epsilon^2 < \epsilon^2$ and $\text{Sf}(B) = O(\epsilon)\ \text{OPT}(I)$ (B contains M and part of M_{i_0}).

Now, FFDW packs A into the $\text{OPT}(I)$ added strips of size $O(\epsilon) \times 1$ while NFDH does the work for the rectangles in B. ∎

LEMMA 3.6. *Consider the two-dimensional NFDH heuristic applied to rectangles in S. When NFDH can not place any other rectangle in a bin of size $a \times b$ then the total unused space in that bin is no more than:*

$$O(\epsilon'\epsilon^2)(a+b)$$

Proof. The result is very similar to Lemma 2.2 and to a result in [2]. ∎

We are now ready to give the overall analysis of the algorithm.

Proof of Theorem 1.2. Consider an input list I and let I' be the rounded input. From Lemma 3.2 we know that there is a packing of I' into no more than $\text{OPT}(I)$ bins of size $(1 + O(\epsilon)) \times (1 + O(\epsilon))$. Consider then the optimal packing in such bins for I' satisfying the conditions of Lemma 3.2. By Lemma 3.3 we know that in such packing all cells as in Lemma 3.3 intersect either an rectangle in L, H or V but not two of them. In other words in an optimal packing of I' satisfying the conditions of Lemma 3.2 each cell is labeled either V, H, or i for $i = 1, \ldots, C$ (where C was the number of distinct large rectangles); or will have no label at all.

Clearly, our algorithm will eventually guess a labeling of the cells such that labeled cell in the optimal packing have the same label. At that point the algorithm will find a feasible packing of all rectangles in L and almost all rectangles in H and V. By Lemmas 3.4 and 3.5 the unpacked rectangles M_{i_0}, M and M' are of small surface and they can be packed in the extra space added in step (5) of the algorithm.

It only remains to see that the small rectangles S will be successfully packed by NFDH. We prove that is not possible that in step (4) a new bin is opened if already $\text{OPT}(I)\ (1+O(\epsilon)) \times (1+O(\epsilon))$ bins have been used. We do this by a surface argument. Suppose, by contradiction, that such new bin is opened in step (4). At this step we distinguish four types of $\epsilon'\epsilon \times \epsilon'\epsilon$ cells: the ones completely filled with a rectangle in L, the ones filled with only rectangles in S, the ones filled only with rectangles in H or V, and the ones partly filled with rectangles in H or V and partly with rectangles in S. By the arguments in Lemmas 3.4 and 3.6 all cells are almost filled Namely a fraction $(1 - O(\epsilon))$ of their surface is filled. Overall this implies that a fraction $(1 - O(\epsilon))$ of the first $\text{OPT}(I)$ bins is filled, and then the total surface that has been filled in the first $\text{OPT}(I)$ $(1 + O(\epsilon)) \times (1 + O(\epsilon))$ squares is at least

$$(1 - O(\epsilon))\ \text{OPT}(I)(1 + O(\epsilon))^2 > \text{OPT}(I).$$

Where the inequality follows by choosing the right constants.

4 Concluding Remarks and Related Problems

Our algorithms are also related to the optimization version of the following interesting question posed by Moser [12]:

Determine the smallest number x such that any system of squares with total area 1 may be parallely packed into a rectangle of area x.

Bounds for this problems have been obtained by Kleitman and Krieger [8] and by Novotny [11]. Note that, although we do not obtain bounds for this problem. Our algorithm can easily be adapted to obtain a PTAS for the following very related optimization problem. This problem has many applications in computer science, for example in VLSI design:

Given a system of n rectangles with total area 1, determine the smallest number x such that all n rectangles may be parallely packed into a rectangle R of area x.

Let us I denote the instance. For each rectangle, let $a_i \in (0, 1]$ denote its width and $b_i \in (0, 1]$ its height; the total surface of the instance, denoted by $\text{Sf}(I)$, equals 1. To solve this problem note first that the techniques in the paper can be easily adapted to solve the underlying decision problem: Given n rectangles with sides $a_i, b_i \in (0, 1]$, given a rectangle R of size $a \times b$ and given $\delta > 0$, determine if all n rectangles can be packed in a rectangle of size $[(1+\delta)a+\delta] \times [(1+\delta)b+\delta]$ or they cannot be packed in R.

Consider a given $\epsilon > 0$, and denote by $\alpha = \min\{\max_{i=1,\ldots,n} a_i, \max_{i=1,\ldots,n} b_i\}$. We distinguish two cases:

(i) $\alpha > \epsilon$. In this case, we ensure that the sides of the minimum area rectangle in which all n items can be

packed are at least ϵ. It is then enough to solve the decision problem, taking $\delta = \epsilon^2$, for all rectangles R such that:

- Their lower-left corner is $(0,0)$ and their upper-right corner is on or below the curve $xy = 4\mathrm{Sf}(I) = 4$ (since the NFDH shelf packing heuristic can always pack I in a square of area $4\mathrm{Sf}(I)$).
- The coordinates of their upper-right corner are at least ϵ.
- Their sides length are integer multiples of ϵ^2.

The number of such rectangles can be evaluated as:

$$\begin{aligned}\int_{\epsilon}^{\mathrm{Sf}(I)/\epsilon} \frac{4\mathrm{Sf}(I)}{x}dx \cdot \frac{1}{\epsilon^4} &= \frac{4\mathrm{Sf}(I)}{\epsilon^4}\ln\left(\frac{\mathrm{Sf}(I)}{\epsilon^2}\right) \\ &= \frac{4}{\epsilon^4}\ln\left(\frac{1}{\epsilon^2}\right) \\ &< \frac{8}{\epsilon^4}\ln\left(\frac{1}{\epsilon}\right).\end{aligned}$$

Which is constant. Therefore by solving the decision problem for a constant number of rectangles, we can choose the one with minimum area. Since in this case a and b are guaranteed to be large, $[(1+\delta)a+\delta] \leq (1+2\epsilon)a$ and $[(1+\delta)b+\delta] \leq (1+2\epsilon)b$. Therefore the best rectangle is guaranteed to have area within a factor of $(1+O(\epsilon))$ that of the optimal one.

(ii) $\alpha \leq \epsilon$. In this case either all rectangles are flat or all are narrow. Without loss of generality assume they are all narrow. We will pack all rectangles using the FFDH heuristic for strip packing in a strip of width $\sqrt{\epsilon}$. A slight adaptation of the following theorem – shown in [2] – proves that all items can be placed in a rectangle of area roughly equal to $\mathrm{Sf}(I)$.

THEOREM 4.1. *In all rectangles in have width less than or equal to δ, then the height $FFDH(I)$ achieved by the FFDH heuristic (on a strip of width 1) satisfies:*

$$FFDH(I) \leq (1+\delta)Sf(I) + 1.$$

In our case the width of the strip is $\sqrt{\epsilon}$ and then $FFDH$ finds a packing in a rectangle of size:

$$\sqrt{\epsilon} \times \left[(1+\sqrt{\epsilon})\frac{\mathrm{Sf}(I)}{\sqrt{\epsilon}} + 1\right].$$

The volume of such rectangle is then

$$(1+\sqrt{\epsilon})\mathrm{Sf}(I) + \sqrt{\epsilon} = (1+2\sqrt{\epsilon}).$$

Again the result follows in this case.

An interesting open question is to generalize Theorem 1.2 to higher dimensional rectangle packing. This does not seems to be a trivial task since a higher dimensional version of strip packing will be needed (and the hardness result in [1] implies that there is no APTAS for three-dimensional strip packing). Another open problem is to determine the approximability of finding a packing of a set of rectangles with $\min(a_i, b_i) > 1/3$ and $\max(a_i, b_i) > 1/2$, that uses the minimum number of unit squares. In this case our rounding techniques do not work. The following example illustrates this fact. Suppose we are given $4n$ rectangles: $2n$ of them with height a little larger than $1/2$ and width a little smaller than $1/2$, and $2n$ with width a little larger than $1/2$ and height a little smaller than $1/2$. Can we decide in polynomial time if they all fit in n unit squares? Even more, Can we decide if they fit in, say, $1.1n$ unit squares? This second question is related to finding hardness of approximation results for rectangle packing. As mentioned before, Bansal and Sviridenko [1] showed that unless $P = NP$ there is no APTAS for the classic two dimensional bin packing problem. Their work partially solves this question and gives good evidence that stronger inapproximability results can be obtained.

Acknowledgments. The authors would like to thank David Johnson for many useful comments and suggestions regarding the presentation of the paper and for pointing out some minor errors.

References

[1] N. Bansal and M. Sviridenko (2004). New approximability and inapproximability results for 2-dimensional bin packing. *These Proceedings.*

[2] E.G. Coffman, Jr., M. R. Garey, D. S. Johnson, and R. E. Tarjan (1980). Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM Journal on Computing* 9, 808-826.

[3] A. Caprara (2002). Packing 2-Dimensional Bins in Harmony, In *Proceeding of the 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS'02).*

[4] A. Caprara, A. Lodi, and M. Monaci (2003). Fast Approximation Schemes for the Two-Stage, Two-Dimensional Bin Packing Problem. Research Report OR/03/6 DEIS.

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

[6] C.E. Ferreira, F.K. Miyazawa, and Y. Wakabayashi (1999). Packing squares into squares. *Pesquisa Operacional* 19, 349-355.

[7] C. Kenyon and E. Remila (2000). A near optimal solution to a two-dimensional cutting stock problem. *Mathematics of Operations Research* 25, 645-656.

[8] D.J. Kleitman and M. Krieger (1975). An optimal bound for two dimensional packing. In *Proceeding of the 16th Annual IEEE Symposium on Foundations of Computer Science (FOCS'75)*.

[9] Y. Kohayakawa, F.K. Miyazawa, P. Raghavan, and Y. Wakabayashi (2001). Multidimensional Cube Packing, In *Proceedings of the Brazilian Symposium on Graphs, Algorithms and Combinatorics (GRACO 2001)*.

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

[11] P. Novotny (1996). On packing of squares into a rectangle. *Archivum Mathematicum* 32, 75-83.

[12] L. Moser (1965). Poorly formulated unsolved problems is combinatorial geometry. Mimeographed.

[13] S.S. Seiden and R. van Stee (2003). New Bounds for Multidimensional Packing, *Algorithmica* 36, 261-293.

[14] V. Vazirani (2001). Approximation Algorithms, Springer-Verlag, Berlin.